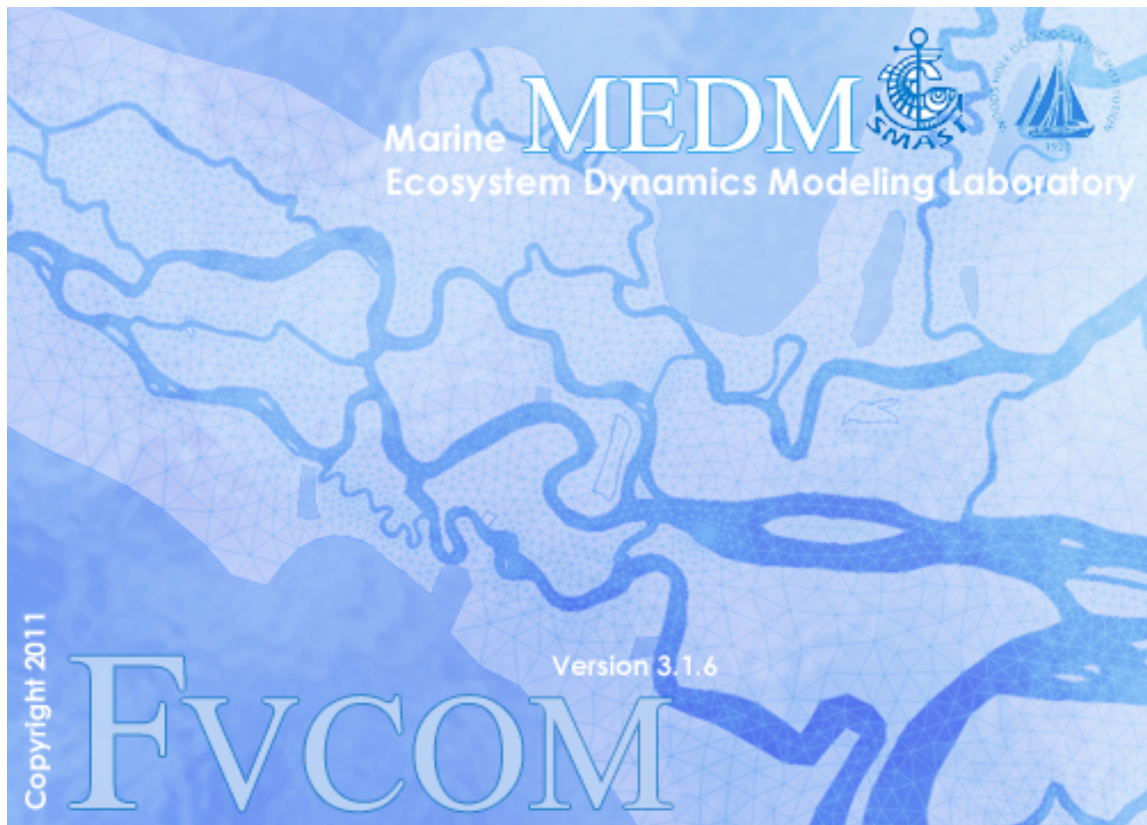# An Unstructured Grid, Finite-Volume Community Ocean Model

FVCOM User Manual

Changsheng Chen[1], Robert C. Beardsley[2] , Geoffrey Cowles[1], Jianhua Qi[1], Zhiqang Lai[1], Guoping Gao[1], David Stuebe[1], Hedong Liu[1], Qichun Xu[1], Pengfei Xue[1], Jianzhong Ge[1], Song Hu[1], Rubao Ji[2], Rucheng Tian[1], Haosheng Huang[1], Lunyu Wu[1], Huichan Lin[1], Yunfang Sun[1] and Liuzhi Zhao[1]

[1]Department of Fisheries Oceanography, School for Marine Science and Technology
University of Massachusetts-Dartmouth, New Bedford, MA 02744

[2]Department of Physical Oceanography
Woods Hole Oceanographic Institution, Woods Hole MA 02543

Fourth Edition

Table of Content

Preface

FVCOM is a prognostic, unstructured-grid, Finite-Volume, free-surface, three-dimensional (3-D) primitive equations Community Ocean Model developed originally by Chen et al. (2003a). The current version of FVCOM is fully coupled ice-ocean-wave-sediment-ecosystem model system with options of various turbulence mixing parameterization, generalized terrain-following coordinates, data assimilation schemes, and wet/dry treatments with inclusion of dike and groyne structures under hydrostatic or non-hydrostatic approximation. FVCOM solves the governing equations on Cartesian or spherical coordinates in integral form by computing fluxes between non-overlapping horizontal triangular control volumes. Either mode-split or semi-implicit schemes can be selected. This finite-volume approach combines the best of finite-element methods (FEM) for geometric flexibility and finite-difference methods (FDM) for simple discrete structures and computational efficiency. This numerical approach also provides a much better representation of mass, momentum, salt, and heat conservation in coastal and estuarine regions with complex geometry. The conservative nature of FVCOM in addition to its flexible grid topology and code simplicity make FVCOM ideally suited for interdisciplinary application in the coastal ocean.

The initial development of FVCOM was started by a team effort led by C. Chen in 1999 at the University of Georgia (UGA) with support from the Georgia Sea Grant College Program. C. Chen, H. Liu, and R. C. Beardsley developed the first version of FVCOM at designing to simulate the 3-D currents and transport within an estuary/tidal creek/inter-tidal salt marsh complex. The first manuscript about this new model was submitted to Journal of Atmospheric and Oceanic Technology in 2000 and published in 2003. That was the first paper of FVCOM. In 2001, C. Chen moved to the School of Marine Science and Technology at the University of Massachusetts-Dartmouth (SMAST/UMASS-D) and established the Marine Ecosystem Dynamics Modeling (MEDM) Laboratory where work on FVCOM has continued with funding from several sources including the NASA and NOAA-funded SMAST fishery program led by Brian Rothschild, the NSF/NOAA US GLOBEC/Georges Bank Program. Led by C. Chen and R. C. Beardsley (Woods Hole Oceanographic Institution-WHOI), the model development team with members of H. Liu, T. Wang completed the original structure of FVCOM and conducted a series of model validation experiments. G. Cowles joined the MEDM group as postdoctoral researcher in 2003 and lead the conversion of FVCOM to Fortran 90/95, modularized the coding structure,

and added the capability for parallel computation. The first FVCOM User Manual was published in 2004 together with a release of FVCOM v2.4. Since then, many new modules were developed by the FVCOM team members including J. Qi, H. Huang, Q. Xu, Z. Lai, P. Xue, D. Stuebe and R. Tian. The second FVCOM User Manual came out in 2006 with a release of FVCOM v2.6. D. Stuebe implemented a new code structure to improve the efficiency of inter-node data exchange and model input and output writing under parallel computational environments, and J. Qi continued to complete his work after he left. D. Stuebe also implemented the visualization software "ViSiT" into FVCOM, which can monitor the model performance during the model run. This new code structure was the origin of FVCOM v3.0.

FVCOM was developed and upgraded by a team effort with numerous contributions from scientists and graduate students at MEDM/SMAST/UMASSD. FVCOM v3.1.6 includes many new modules listed below.

UG-CICE: An unstructured grid sea ice model converted from the structured grid CICE. This model was developed as one component of G. Gao's Ph.D. thesis research supervised by C. Chen, R. C. Beardsley and A. Proshutinsky (Gao, 2011). G. Gao and J. Qi wrote the code, with support from C. Chen and R. Beardsley in numerical algorithms. The validation was carried out by G. Gao, C. Chen and R. C. Beardsley (Gao et al., 2011).

FVCOM-NH: A non-hydrostatic version of FVCOM developed as one component of Z. Lai's Ph.D. thesis research (Lai, 2009) under supervision of C. Chen, G. Cowles and R. C. Beardsley. Z. Lai wrote the code, with support from C. Chen in discrete algorithms and C. Gowles in matrix solvers. The validation experiments were carried out by Z. Lai, C. Chen, G. Cowles and R. C. Beardsley (Lai et al., 2010a,b).

FVCOM-SWAVE: An unstructured grid version of surface wave model-SWAN developed by J. Qi, C. Chen and R. C. Beardsley (Qi et al., 2008). J. Qi wrote the code with supports from C. Chen in discrete algorithms and Z. Lai and G. Cowles in matrix solvers. J. Qi, C. Chen and R. C. Beardsley did the validation experiments and also applied this model to the Gulf of Maine region. Coupling SWAVE into FVCOM was initialized by A. Wu as a visiting student at SMAST/UMASSD for his Ph.D. thesis research under supervision of C. Chen (Wu et al., 2011). J. Qi and Q. Xu re-organized, modified and re-debugged the code to have more flexible setup and C. Chen, J. Qi and Q. Xu improved the coupling algorithms to improve the reality, stability and accuracy of wave-current interaction code. A fully current-wave-sediment module in

FVCOM was also initialized by A. Wu and J. Ge (a visiting student from East China Normal University). J. Qi and Q. Xu re-organized the code and updated to FVCOM v3.1.6 or up.

FVCOM-GEM: A generalized biological module coupled with FVCOM, which allow users to select either a pre-built biological model (such as a NPZ, NPZD, NPZDB, etc) or construct their own biological model using the pre-refined pool of biological variables and parameterization functions. This model was developed by C. Chen, R. Tian, J. Qi and R. Ji. C. Chen and R. Tian constructed the math equations. C. Chen designed the code structure and J. Qi initialized the code development. R. Tian, J. Qi and R. Ji wrote and tested the first online code, and J. Qi upgraded the code to FVCOM v3.1.6.

FVCOM-SED: An unstructured-grid version of the USGS structured-grid community sediment model developed by G. Cowles. With Warner's support, G. Cowles converted and updated the USGS code to the unstructured grid version under the FVCOM framework.

FVCOM-WQM: A water quality model based on the EPA Water quality Analysis Simulation Program (WASP) implemented into FVCOM by J. Qi and C. Chen.

UG-CE-QUAL-ICM: An unstructured-grid finite-volume version of the Army Corp of Engineers water quality model CE-QUAL-ICM. This code was originally developed by J. Qi and C. Chen and modified and validated by scientists at DOE-Pacific Northwest National Laboratory.

UG-RCA: An unstructured grid version of the water quality model (RCA) developed by J. Qi, C. Chen, and R. Tian. J. Qi and C. Chen converted RCA to UG-RCA. R. Tian conducted the validation experiments. L. Zhao developed a pre-processing package to make it easy in using UG-RCA.

Dike-Groyne Module: A module to treat vertical straight walls above and below the sea levels. The algorithm was derived by C. Chen at SMAST/UMASSD, and the parallelized code was written by J. Qi and J. Ge. J. Ge conducted a series of validation experiments (Ge et al. 2011).

FVCOM-DYE: A module used to trace the dye online. This module was developed by Q. Xu and S. Hu.

Semi-implicit Solver: A semi-implicit solver was implemented into FVCOM as an option for time integration by Z. Lai and C. Chen.

Multi-domain Nesting Module: An input- and output-module designed for multi-domain nesting. A one-way nesting module was originally written by P. Xue when he and Chen applied

FVCOM Kalman Filter to conduct The Observing System Simulation Experiments (OSSEs) in Nantucket Sound. This module was modified by D. Stuebe and implemented into the version 3.0 of FVCOM. D. Stuebe developed a one-way nesting module to multi-domain nesting. G. Gao and Y. Sun developed a pre-processing program to created a nesting boundary file that allows the global-FVCOM nests the regional domain FVCOM with remaining the regional domain tidal forcing boundary. S. Hu developed a one-way nesting module to use the global HYCOM output to drive the regional FVCOM. J. Qi re-organized all nesting programs and upgraded them in FVCOM v3.1.6. and Q. Xu and J. Qi have tested these modules in applications to Scituate inundation applications and Air France 477 search.

SST/SSH Assimilation Modules: The nudging SST/TS assimilation module was initialized by H. Liu and modified by Q. Xu and J. Qi. Q. Xu and J. Qi coded the OI module. Z. Lai modified this module for the SSH data assimilation and implemented PWP mixed layer model in SST/TS data assimilation. Q. Xu, J. Qi, Z. Lai and G. Gao upgraded the modules to FVCOM v3.1.6.

Kalman Filter Assimilation Module: This module was originally developed by a team effort led by C. Chen at SMAST/UMASSD, P. Rizzoli at MIT and R. C. Beardsley at WHOI. The team members include P. Xue, Q. Xu, Z. Lai and J. Qi at SMAST/UMASSD and S. Lyu and J. Wei at MIT. P. Xue upgraded this package by adding a Signular Evolutive Interpolated Kalman filter (SEIK) and also re-constructed the package to FVCOM v3.1.6. The source code of SEIK code is provided by Dr. Lars Nerger at AWI, German.

In an early stage in the FVCOM development, D. Chapman (WHOI) gave many valuable suggestions and comments on the code structure and model validation. He spent significant times on working together with Chen on debugging the code and validation experiments, although those results were never published. J. Qi has been in charge of updating the codes and validation tests. He has devoted much time to combining all modifications into the updated version.

 We greatly appreciate all the users who have made great contributions to the code validation studies and applications to various coastal and regional ocean environments. Bugs reported by users have been very helpful and have contributed to FVCOM's reliability and accurate. The development of FVCOM has benefited from users' comments and suggestions.

As the FVCOM development team leader, Changsheng Chen reserves all rights of this product. The University of Massachusetts-Dartmouth owns the copyright of the software of this

model. All copyrights are reserved. Unauthorized reproduction and distribution of this program are expressly prohibited. This program is permitted for use in non-commercial academic research and education. The commercial use may be subject to a fee. Modification is not encouraged for users who do not have a deep understanding of the code structures and finite-volume numerical methods used in FVCOM. Contributions made to correcting and modifying the program will be credited, but not affect copyrights. For public use, all users should name this model as "**FVCOM**". In any publications with the use of FVCOM, acknowledgement must be included.

FVCOM Software Users' License Agreement

All users should read this agreement carefully. A user, who receives any version of the source code of FVCOM, must accept all the terms and conditions of this agreement and also agree that this agreement is like any written negotiated agreement signed by you. You may be required to have another written agreement directly with Dr. Changsheng Chen at SMAST/UMASS-D and Dr. Robert C. Beardsley at WHOI

The Finite-Volume Coastal Ocean Model ("FVCOM") source code has been developed in the Marine Ecosystem Dynamics Modeling Laboratory led by Dr. C. Chen at the University of Massachusetts – Dartmouth (UMASS-D) in collaboration with Dr. R. Beardsley at the Woods Hole Oceanographic Institution. All copyrights to the FVCOM code are reserved. Unauthorized reproduction and redistribution of this code are expressly prohibited except as allowed in this License.

A. Permitted Use and Restrictions on Redistribution

The user agrees that he/she will use the FVCOM source code, and any modifications to the FVCOM source code that the user may create, solely for internal, non-commercial purposes and shall not distribute or transfer the FVCOM source code or modifications to it to any person or third parties not participating in their primary research project without prior written permission from Dr. Chen. The term "non-commercial," as used in this End User License Agreement, means academic or other scholarly research which (a) is not undertaken for profit, or (b) is not intended to produce work, services, or data for commercial use.

B. Mandatory Participation in the FVCOM Community

The user agrees to openly participate in the FVCOM community through three primary mechanisms. These are (*a*) reporting code bugs and problems, (*b*) sharing major modifications made to the code, and (*c*) contributing to an open and ongoing discussion of model deficiencies, needed improvements and additions, and major successes. (Contact Drs. C. Chen, G. Cowles, or R. Beardsley). These mechanisms are intended to benefit the entire FVCOM user community through quick notification of code problems, possible solutions, major code improvements, and, in general, the further development of the FVCOM source code and the associated software tools needed to process, visualize and interpret FVCOM model output.

C. FVCOM Validation

The user agrees to inform Dr. Chen about any FVCOM model validation test case conducted by the user before formal publication of the test case results. This step is intended to minimize potential errors in gridding, model setup, boundary conditions and coding that could contribute to poor FVCOM performance in the validation test case. There is no intent here to exercise any prior restraint on publication.

D. Publication of FVCOM Results

The user agrees to acknowledge FVCOM in any publications resulting from the use of the FVCOM source code. The user agrees to use the name "FVCOM" to refer to the model.

Chapter 1: Introduction

Throughout much of the world oceans, the inner continental shelves and estuaries are characterized by barrier island complexes, inlets, and extensive intertidal salt marshes. Such an irregularly shaped ocean-land margin system presents a serious challenge for oceanographers involved in model development even though the governing equations of ocean circulation are well defined and numerically solvable in terms of discrete mathematics. Two numerical methods have been widely used in ocean circulation models: (1) the finite-difference method (Blumberg and Mellor, 1987; Blumberg, 1994; Haidvogel et al., 2000) and (2) the finite-element method (Lynch and Naimie, 1993; Naimie, 1996). The finite-difference method is the most basic discrete scheme and has the advantage of computational and coding efficiency. Introducing an orthogonal or non-orthogonal curvilinear horizontal coordinate transformation into a finite-difference model can provide adequate boundary fitting in relatively simple coastal regions but these transformations are incapable of resolving the highly irregular inner shelf/estuarine geometries found in many coastal areas (Blumberg 1994; Chen et al. 2001; Chen et al. 2004a). The greatest advantage of the finite-element method is its geometric flexibility. Triangular grid meshes of arbitrary spatially dependent size are commonly used in this method, and can provide an accurate fitting of the irregular coastal boundary. The P-type Finite-Element Method (Maday and Patera, 1988) or Discontinuous Galerkin Method (Reed and Hill, 1973; Cockburn et al., 1990) has recently been applied to ocean and have shown promise in improving both computational accuracy and efficiency.

We have developed a 3-D unstructured-grid, free-surface, primitive equation, Finite-Volume Coastal Ocean circulation Model (called FVCOM) (Chen et al. 2003a; Chen et al. 2004b, Chen et al., 2006a-c, Chen et al., 2007). Unlike the differential form used in finite-difference and finite-element models, FVCOM discretizes the integral form of the governing equations. Since these integral equations can be solved numerically by flux calculation (like those used in the finite-difference method) over an arbitrarily sized triangular mesh (like those used in the finite-element method), the finite-volume approach is better suited to guarantee mass conservation in both the individual control element and the entire computational domain. From a technical point of view, FVCOM combines the best attributes of finite-difference methods for simple discrete coding and

computational efficiency and finite-element methods for geometric flexibility. This model has been successfully applied to study several estuarine and shelf regions that feature complex irregular coastline and topographic geometry, including inter-tidal flooding and drying (see http://fvcom.smast.umassd.edu for descriptions of these initial applications).

This manual is provided to help users to 1) understand the basic discrete structure and numerical methods used in FVCOM and 2) learn how to use the model for their own applications. Detailed instructions are given for all steps (e.g., grid generation, model input and output, compilation, parallel computation, etc.). Several experiments are included to provide new users with simple examples of model setup and execution

The remaining chapters are organized as follows. Chapter 2: the model formulation; Chapter 3: the finite-volume discrete method; Chapter 4: the non-hydrostatic FVCOM; Chapter 5: the external forcings; Chapter 6: the open boundary treatments; Chapter 7: the dike and groyne module; Chapter 8: the sediment module; Chapter 9: the surface wave model-SWAVE; Chapter 10: the sea ice model-UG-CICE; Chapter 11: the biological modules, Chapter 12: the tracer-tracking module; Chapter 13: the 3-D Lagrangian particle tracking; Chapter 14: the 4-D data assimilation methods; Chapter 15: the code parallelization; Chapter 16: the model coding description and general information; Chapter 17: the model installation; Chapter 18: the model setup; Chapter 19: examples of model applications, and Chapter 20: an example of the unstructured grid generation.

Users should be aware that this manual is only useful for the current version of FVCOM. FVCOM is in continual testing and improvement by a SMAST/UMASSD-WHOI effort led by Changsheng Chen and Robert C. Beardsley. Some very recent modifications may not have been included in this manual. If users find any inconsistency between this manual and the FVCOM code, it is likely to be due to a typo in the manual. Please report any problems with this manual as well as suggestions for improvement, so that future versions can be enhanced.

Chapter 2: The Model Formulation

2.1 The Primitive Equations in Cartesian Coordinates

Under absence of snow and ice, the governing equations consist of the following momentum, continuity, temperature, salinity, and density equations:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv = -\frac{1}{\rho_o}\frac{\partial(p_H + p_a)}{\partial x} - \frac{1}{\rho_o}\frac{\partial q}{\partial x} + \frac{\partial}{\partial z}(K_m\frac{\partial u}{\partial z}) + F_u \tag{2.1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu = -\frac{1}{\rho_o}\frac{\partial(p_H + p_a)}{\partial y} - \frac{1}{\rho_o}\frac{\partial q}{\partial y} + \frac{\partial}{\partial z}(K_m\frac{\partial v}{\partial z}) + F_v \tag{2.2}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho_o}\frac{\partial q}{\partial z} + \frac{\partial}{\partial z}(K_m\frac{\partial w}{\partial z}) + F_w$$

$$(2.3)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.4}$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} + w\frac{\partial T}{\partial z} = \frac{\partial}{\partial z}(K_h\frac{\partial T}{\partial z}) + F_T \tag{2.5}$$

$$\frac{\partial S}{\partial t} + u\frac{\partial S}{\partial x} + v\frac{\partial S}{\partial y} + w\frac{\partial S}{\partial z} = \frac{\partial}{\partial z}(K_h\frac{\partial S}{\partial z}) + F_S \tag{2.6}$$

$$\rho = \rho(T, S, p) \tag{2.7}$$

where x, y, and z are the east, north, and vertical axes in the Cartesian coordinate system; u, v, and w are the x, y, z velocity components; T is the temperature; S is the salinity; ρ is the density; p_a is the air pressure at sea surface; p_H is the hydrostatic pressure; q is the non-hydrostatic pressure; f is the Coriolis parameter; g is the gravitational acceleration; K_m is
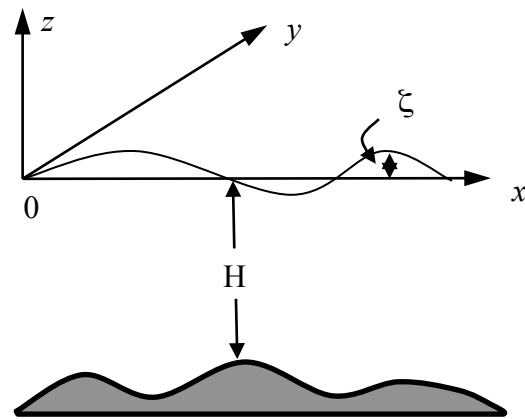
Fig. 2.1: Illustration of the orthogonal coordinate system: x: eastward; y: northward; z: upward.

the vertical eddy viscosity coefficient; and K_h is the thermal vertical eddy diffusion

coefficient. F_u, F_v, F_T and F_s represent the horizontal momentum, thermal, and salt diffusion terms. The total water column depth is $D = H + \zeta$, where H is the bottom depth (relative to $z = 0$) and ζ is the height of the free surface (relative to z = 0). $p = p_a + p_H + q$ is the total pressure, in which the hydrostatic pressure p_H satisfies

$$\frac{\partial p_H}{\partial z} = -\rho g \Rightarrow p_H = \rho_o g \zeta + g \int_z^0 \rho \, dz' \tag{2.8}$$

The surface and bottom boundary conditions for temperature are:

$$\frac{\partial T}{\partial z} = \frac{1}{\rho c_p K_h} [Q_n(x,y,t) - SW(x,y,\zeta,t)], \quad \text{at } z = \zeta \, (x, y, t) \tag{2.9}$$

$$\frac{\partial T}{\partial z} = -\frac{A_H \tan \alpha}{K_h} \frac{\partial T}{\partial n}, \quad \text{at } z = -H(x, y) \tag{2.10}$$

where $Q_n(x,y,t)$ is the surface net heat flux, which consists of four components: net downward shortwave radiation, net downward longwave radiation, sensible, and latent fluxes, $SW(x,y,\zeta,t)$ is the shortwave flux incident at the sea surface, and c_p is the specific heat of seawater. A_H is the horizontal thermal diffusion coefficient, α is the slope of the bottom bathymetry, and n is the horizontal coordinate shown in Figure 2.2 (Pedlosky, 1974; Chen et al., 2004b).
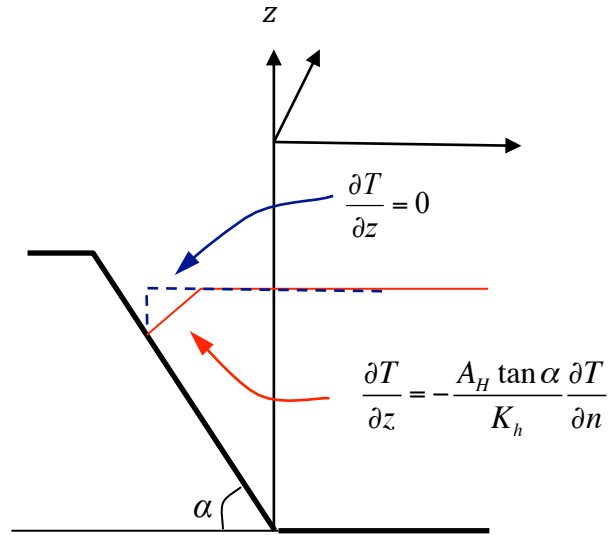
Fig. 2.2: Illustration of the bottom boundary condition for temperature over a sloping bottom.

The longwave, sensible and latent heat fluxes are assumed here to occur at the ocean surface, while the downward shortwave flux $SW(x,y,z,t)$ is approximated by:

$$SW(x,y,z,t) = SW(x,y,\zeta,t)[R e^{\frac{z}{a}} + (1 - R) e^{\frac{z}{b}}] \tag{2.11}$$

where a and b are attenuation lengths for longer and shorter (blue-green) wavelength components of the shortwave irradiance, and R is the percent of the total flux associated with the longer wavelength irradiance. This absorption profile, first suggested by Kraus (1972), has been used in numerical studies of upper ocean diurnal heating by Simpson and Dickey (1981a, b) and others. The absorption of downward irradiance is included in the temperature (heat) equation in the form of

$$\hat{H}(x,y,z,t) = \frac{\partial SW(x,y,z,t)}{\partial z} = \frac{SW(x,y,0,t)}{\rho c_p}[\frac{R}{a}e^{\frac{z}{a}} + \frac{1-R}{b}e^{\frac{z}{b}}] \qquad (2.12)$$

This approach leads to a more accurate prediction of near-surface temperature than the flux formulation based on a single wavelength approximation (Chen et al., 2003b).

The surface and bottom boundary conditions for salinity are:

$$\frac{\partial S}{\partial z} = 0, \quad \text{at } z = \zeta\,(x,\,y,\,t) \qquad (2.13)$$

$$\frac{\partial S}{\partial z} = \frac{A_H \tan\alpha}{K_h}\frac{\partial S}{\partial n}, \quad \text{at } z = -H(x,\,y) \qquad (2.14)$$

The surface and bottom boundary conditions for u, v, and w are:

$$K_m(\frac{\partial u}{\partial z}, \frac{\partial v}{\partial z}) = \frac{1}{\rho_o}(\tau_{sx}, \tau_{sy}), w = \frac{\partial\zeta}{\partial t} + u\frac{\partial\zeta}{\partial x} + v\frac{\partial\zeta}{\partial y} + \frac{E-P}{\rho}, \qquad \text{at } z = \zeta\,(x,\,y,\,t) \qquad (2.15)$$

$$K_m(\frac{\partial u}{\partial z}, \frac{\partial v}{\partial z}) = \frac{1}{\rho_o}(\tau_{bx}, \tau_{by}), w = -u\frac{\partial H}{\partial x} - v\frac{\partial H}{\partial y} + \frac{Q_b}{\Omega}, \qquad \text{at } z = -H(x,\,y) \qquad (2.16)$$

where (τ_{sx}, τ_{sy}) and $(\tau_{bx}, \tau_{by}) = C_d\sqrt{u^2 + v^2}(u,v)$ are the x and y components of surface wind and bottom stresses, Q_b is the groundwater volume flux at the bottom and Ω is the area of the groundwater source. The drag coefficient C_d is determined by matching a logarithmic bottom layer to the model at a height z_{ab} above the bottom, *i.e.*

$$C_d = \max\left(k^2/\ln(\frac{z_{ab}}{z_o})^2, 0.0025\right) \qquad (2.17)$$

where $k = 0.4$ is the von Karman constant and z_o is the bottom roughness parameter.

The kinematic and heat and salt flux conditions on the lateral solid boundary are specified as:

$$v_n = 0; \frac{\partial T}{\partial n} = 0; \frac{\partial S}{\partial n} = 0 \,, \tag{2.18}$$

where v_n is the velocity component normal to the boundary, and n is the coordinate normal to the boundary.

It should be pointed out here that in most popular finite-difference models, the bottom boundary conditions for temperature and salinity are simplified as $\partial T / \partial z = \partial S / \partial z = 0$. One reason for this is the difficulty in the finite-difference method of calculating accurately α and $\partial T / \partial z$ or $\partial S / \partial z$ over an irregular bottom slope. The error caused by inaccurate calculation of these two terms in a finite-difference approach might be larger than their real values. This simplification is generally sound for much of the continental shelf in the coastal ocean where the bottom topography is smooth with small slope, but over the shelf break and continental slope where the bottom slope can be quite large, this simplification can destroy the nature of the dynamics of the bottom boundary layer and result in overestimation of vertical mixing and horizontal and vertical velocities. An example for the importance of the exact expression of the no normal flux condition at the bottom given in (2.10) and (2.14) can be seen in Chen et al. (2007). In the finite-volume approach, the bottom slope and gradients of temperature and salinity for an irregular bottom shape can be directly calculated using a simple Green's theorem. Therefore, FVCOM can provide an accurate tracer flux at the bottom using (2.10) and (2.14). This is one of the advantages for using FVCOM in both coastal and deep ocean applications.

2.2 The Governing Equations in the Terrain-following Coordinate

Define that $\hat{g} = \hat{g}(x, y, r, t)$ is a generalized terrain-following coordinate system, in which x and y are defined as the eastward and northward axes, and r is defined as the vertical axis varying from -1 to 0. r can be specified as a sigma, hybrid or more generalized function (see various choices in Chapter 18). An example for a hybrid coordinate is shown in Fig. 2.3.
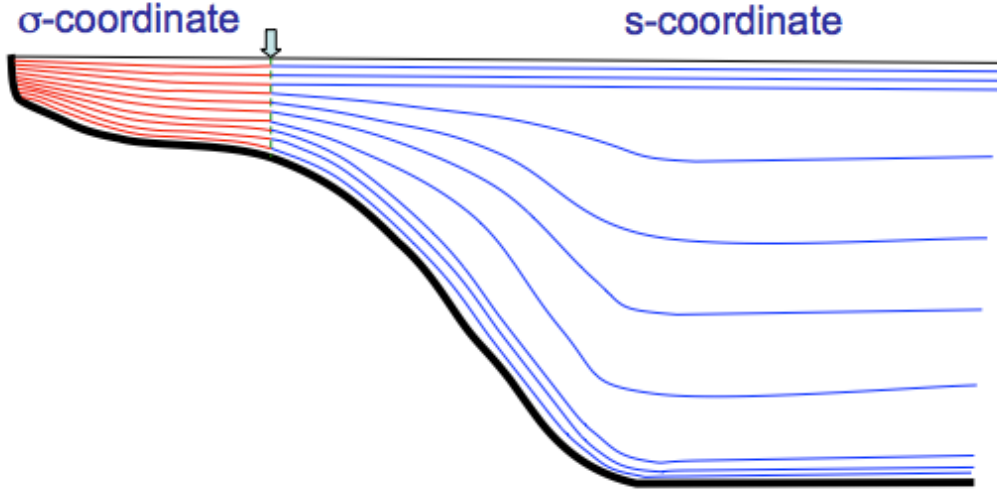
Fig. 2.3: An example of a hybrid coordinate consisting of σ- and *s*-coordinates. The dashed line indicates a transition location of two coordinates at which all layers are uniformly divided in the vertical.

In this generalized terrain-following coordinate system, equations (2.1)-(2.7) can be written as

$$\frac{\partial uJ}{\partial t} + \frac{\partial u^2 J}{\partial x} + \frac{\partial uvJ}{\partial y} + \frac{\partial u\omega}{\partial r} - fvJ = -gJ\frac{\partial \zeta}{\partial x} - \frac{J}{\rho_o}\frac{\partial P_a}{\partial x}$$

$$- \frac{gJ}{\rho_o}[\int_r^0 J(\frac{\partial \rho}{\partial x} + \frac{\partial \rho}{\partial r'}\frac{\partial r'}{\partial x})dr'] - \frac{1}{\rho_o}(\frac{\partial qJ}{\partial x} + \frac{\partial qA_1}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial u}{\partial r}) + JF_u$$

(2.19)

$$\frac{\partial vJ}{\partial t} + \frac{\partial uvJ}{\partial x} + \frac{\partial v^2 J}{\partial y} + \frac{\partial v\omega}{\partial r} + fuJ = -gJ\frac{\partial \zeta}{\partial y} - \frac{J}{\rho_o}\frac{\partial P_a}{\partial y}$$

$$- \frac{gJ}{\rho_o}[\int_r^0 J(\frac{\partial \rho}{\partial y} + \frac{\partial \rho}{\partial r'}\frac{\partial r'}{\partial y})dr'] - \frac{1}{\rho_o}(\frac{\partial qJ}{\partial y} + \frac{\partial qA_2}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial v}{\partial r}) + JF_v$$

(2.20)

$$\frac{\partial wJ}{\partial t} + \frac{\partial uwJ}{\partial x} + \frac{\partial vwJ}{\partial y} + \frac{\partial w\omega}{\partial r} = -\frac{1}{\rho_o}\frac{\partial q}{\partial r} + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial w}{\partial r}) + JF_w$$

(2.21)

$$\frac{\partial uJ}{\partial x} + \frac{\partial vJ}{\partial y} + \frac{\partial uA_1}{\partial r} + \frac{\partial vA_2}{\partial r} + \frac{\partial w}{\partial r} = 0$$

(2.22)

$$\frac{\partial TJ}{\partial t} + \frac{\partial TuJ}{\partial x} + \frac{\partial TvJ}{\partial y} + \frac{\partial T\omega}{\partial r} = \frac{\partial}{\partial r}(\frac{K_h}{J}\frac{\partial T}{\partial r}) + J\hat{H} + JF_T$$

(2.23)

$$\frac{\partial SJ}{\partial t} + \frac{\partial SuJ}{\partial x} + \frac{\partial SvJ}{\partial y} + \frac{\partial S\omega}{\partial r} = \frac{\partial}{\partial r}(\frac{K_h}{J}\frac{\partial S}{\partial r}) + JF_S \qquad (2.24)$$

$$\rho = \rho(T,S,p) \qquad (2.25)$$

where $J = \partial z/\partial r$, A_1 and A_2 are coordinate transformation coefficients defined as $A_1 = J\partial r/\partial x$ and $A_2 = J\partial r/\partial y$.

The transformed vertical velocity ω satisfies the continuity equation in the form of

$$\frac{\partial J}{\partial t} + \frac{\partial uJ}{\partial x} + \frac{\partial vJ}{\partial y} + \frac{\partial \omega}{\partial r} = 0 \qquad (2.26)$$

and the relationship between ω and the true vertical velocity (w) is given as

$$\omega = \frac{1}{J}(w - \frac{\partial \hat{g}}{\partial t} - \vec{v}\cdot\nabla\hat{g}) \qquad (2.27)$$

In the generalized terrain-following coordinate system, the horizontal diffusion terms are defined as:

$$DF_x \approx \frac{\partial}{\partial x}[2A_mH\frac{\partial u}{\partial x}] + \frac{\partial}{\partial y}[A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})] \qquad (2.28)$$

$$DF_y \approx \frac{\partial}{\partial x}[A_mH(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x})] + \frac{\partial}{\partial y}[2A_mH\frac{\partial v}{\partial y}] \qquad (2.29)$$

$$D(F_T,F_S,F_{q^2},F_{q^2l}) \approx [\frac{\partial}{\partial x}(A_hH\frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(A_hH\frac{\partial}{\partial y})](T,S,q^2,q^2l) \qquad (2.30)$$

where A_m and A_h are the horizontal eddy and thermal diffusion coefficients, respectively. According to the argument made by Mellor and Blumberg (1985), the simplification made in (2.28)-(2.30) helps to ensure the validity of the locally 1-D bottom boundary layer simulation in the topographic-coordinate transformation system. In physics, these simplifications are equivalent to the assumption that horizontal diffusion occurs only parallel to the r-layers. It is clear that this simplification can lead to additional vertical mixing in the slope region due to the coordinate transformation, thus making the model-predicted thermoclines too diffusive in the vertical. Questions related to the horizontal diffusion terms and the stability of FVCOM without these terms have been addressed in the FVCOM development. See Chen et al. (2003a) for additional discussion about these simplifications.

The boundary conditions are given as follows. At the surface where r = 0,

$$(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial r}) = \frac{J}{\rho_o K_m}(\tau_{sx}, \tau_{sy}), \; \omega = 0; \tag{2.31}$$

$$\frac{\partial T}{\partial r} = \frac{J}{\rho c_p K_h}[Q_n(x,y,t) - SW(x,y,0,t)], \; \frac{\partial S}{\partial r} = 0 ; \tag{2.32}$$

and at the bottom where r = -1,

$$(\frac{\partial u}{\partial r}, \frac{\partial v}{\partial r}) = \frac{J}{\rho_o K_m}(\tau_{bx}, \tau_{by}), \; \omega = 0; \tag{2.33}$$

$$\frac{\partial T}{\partial r} = -\frac{A_H \tan\alpha}{K_h / J + A_H \tan\alpha \frac{\partial r}{\partial n}} \frac{\partial T}{\partial n}; \; \frac{\partial S}{\partial r} = -\frac{A_H \tan\alpha}{K_h / J + A_H \frac{\partial r}{\partial n} \tan\alpha} \frac{\partial S}{\partial n}. \tag{2.34}$$

2.3 The 2-D (Vertically Integrated) Equations

The sea-surface elevation included in the equations describes the fast moving ($\sqrt{gD}$) long surface gravity waves. In the explicit numerical approach, the criterion for the time step is inversely proportional to the phase speed of these waves. Since the sea-surface elevation is proportional to the gradient of water transport, it can be computed using vertically integrated equations. The 3-D equations then can be solved under conditions with a given sea-surface elevation. In this numerical method, called "mode splitting", the currents are divided into external and internal modes that can be computed using two distinct time steps. This approach is successfully used in POM and ROMS.

The 2-D (vertically integrated) momentum and continuity equations in the generalized terrain-following coordinate system are given as:

$$\frac{\partial \zeta}{\partial t} + \frac{\partial (\bar{u}D)}{\partial x} + \frac{\partial (\bar{v}D)}{\partial y} = 0 \tag{2.35}$$

$$\frac{\partial \bar{u}D}{\partial t} + \frac{\partial \bar{u}^2 D}{\partial x} + \frac{\partial \bar{u}\bar{v}D}{\partial y} - f\bar{v}D - D\bar{F}_u - G_x - \frac{\tau_{sx} - \tau_{bx}}{\rho_o}$$
$$= -gD\frac{\partial \zeta}{\partial x} - \frac{D}{\rho_o}\frac{\partial p_a}{\partial x} - \frac{g}{\rho_o}\int_{-1}^{0}\{J[\int_{r}^{0} J(\frac{\partial \rho}{\partial x} + \frac{\partial \rho}{\partial r'}\frac{\partial r'}{\partial x})dr']\}dr' - \frac{1}{\rho_o}\int_{-1}^{0}(\frac{\partial qJ}{\partial x} + \frac{\partial qA_1}{\partial r})dr' \tag{2.36}$$

$$\frac{\partial \bar{v}D}{\partial t} + \frac{\partial \overline{uv}D}{\partial x} + \frac{\partial \bar{v}^2 D}{\partial y} + f\bar{u}D - D\overline{F}_v - G_y - \frac{\tau_{sy} - \tau_{by}}{\rho_o}$$

$$= -gD\frac{\partial \zeta}{\partial y} - \frac{D}{\rho_o}\frac{\partial p_a}{\partial y} - \frac{g}{\rho_o}\int_{-1}^{0}\{J[\int_{r}^{0}J(\frac{\partial \rho}{\partial y} + \frac{\partial \rho}{\partial r'}\frac{\partial r'}{\partial y})dr']\}dr' - \frac{1}{\rho_o}\int_{-1}^{0}(\frac{\partial qJ}{\partial y} + \frac{\partial qA_2}{\partial r})dr'$$

(2.37)

where G_x and G_y are defined as

$$G_x = \frac{\partial \overline{u}^2 D}{\partial x} + \frac{\partial \overline{uv}D}{\partial y} - D\tilde{F}_x - [\frac{\partial \overline{u^2}D}{\partial x} + \frac{\partial \overline{uv}D}{\partial y} - D\overline{F}_x]$$

(2.38)

$$G_y = \frac{\partial \overline{uv}D}{\partial x} + \frac{\partial \overline{v}^2 D}{\partial y} - D\tilde{F}_y - [\frac{\partial \overline{uv}D}{\partial x} + \frac{\partial \overline{v^2}D}{\partial y} - D\overline{F}_y]$$

(2.39)

and the horizontal diffusion terms are approximately given as

$$D\tilde{F}_x \approx \frac{\partial}{\partial x}[2\overline{A}_m H\frac{\partial \overline{u}}{\partial x}] + \frac{\partial}{\partial y}[\overline{A}_m H(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x})]$$

(2.40)

$$D\tilde{F}_y \approx \frac{\partial}{\partial x}[\overline{A}_m H(\frac{\partial \overline{u}}{\partial y} + \frac{\partial \overline{v}}{\partial x})] + \frac{\partial}{\partial y}[2\overline{A}_m H\frac{\partial \overline{v}}{\partial y}]$$

(2.41)

$$D\overline{F}_x \approx \frac{\partial}{\partial x}\overline{2A_m H\frac{\partial u}{\partial x}} + \frac{\partial}{\partial y}\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}$$

(2.42)

$$D\overline{F}_y \approx \frac{\partial}{\partial x}\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})} + \frac{\partial}{\partial y}\overline{2A_m H\frac{\partial v}{\partial y}} .$$

(2.43)

The overbar "—" denotes the vertically integration. For example, for a given variable ψ,

$$\overline{\psi} = \int_{-1}^{0}\psi\, d\sigma .$$

(2.44)

2.4 The Primitive Equations in the Spherical Coordinate System

FVCOM was originally coded for the local Cartesian coordinate system in which the Coriolis parameter f varies with latitude but the curvature terms due to the spherical shape of the earth were not included in the momentum equations. Therefore, it is suitable for regional applications but not for a basin or global scale study. To make FVCOM flexible for either regional or global application, we have built a spherical coordinate version of FVCOM (Chen *et al.* 2006a, 2009a, Gao et al. 2011).

Consider a spherical coordinate system in which the x (eastward) and y (northward) coordinates are expressed as

$$x = R\cos\phi(\lambda - \lambda_0), \ \ y = R(\phi - \phi_0) \quad (2.43)$$

where R is the earth's radius; λ is longitude; φ is latitude, and λ_0 and φ_0 are the reference longitude and latitude, respectively. The vertical coordinate z is normal to the earth surface and positive in the upward direction. An illustration of this coordinate is shown in Fig. 2.4.
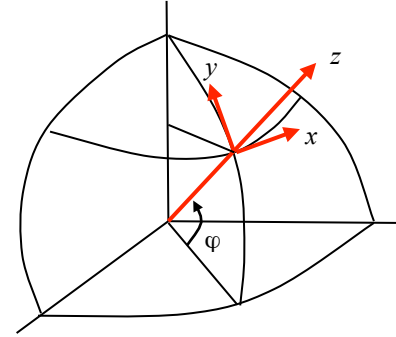
Fig. 2.4: Illustration of the spherical coordinate system.

The three-dimensional (3-D) internal mode flux forms of the governing equations of the motion in the spherical and σ coordinates are given as

$$\frac{\partial u}{\partial t} + \frac{1}{R\cos\phi}[\frac{\partial u^2 J}{\partial \lambda} + \frac{\partial uvJ\cos\phi}{\partial \phi}] + \frac{\partial u\varpi}{\partial \sigma} + \frac{uvJ}{R}\tan\phi - \frac{wuJ}{R} - fvJ - JF_u$$
$$= -\frac{gJ}{R\cos\phi}\frac{\partial \zeta}{\partial \lambda} - \frac{J}{\rho_o R\cos\phi}\frac{\partial p_a}{\partial \lambda} - \frac{gJ}{\rho_o R\cos\phi}[\int_r^0 J(\frac{\partial \rho}{\partial \lambda} + \frac{\partial \rho}{\partial r'}\frac{\partial r}{\partial \lambda})dr'] \quad (2.45)$$
$$-\frac{1}{\rho_o}(\frac{1}{R\cos\phi}\frac{\partial qJ}{\partial \lambda} + \frac{\partial qA_1}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial u}{\partial r})$$

$$\frac{\partial v}{\partial t} + \frac{1}{R\cos\phi}[\frac{\partial uvJ}{\partial \lambda} + \frac{\partial v^2 J\cos\phi}{\partial \phi}] + \frac{\partial v\varpi}{\partial \sigma} + \frac{u^2 J}{R}\tan\phi - \frac{wvJ}{R} + fuJ - JF_v$$
$$= -\frac{gJ}{R}\frac{\partial \zeta}{\partial \phi} - \frac{J}{\rho_o R}\frac{\partial p_a}{\partial \phi} - \frac{gJ}{\rho_o R}[\int_r^0 J(\frac{\partial \rho}{\partial \phi} + \frac{\partial \rho}{\partial r'}\frac{\partial r}{\partial \phi})dr'] \quad (2.46)$$
$$-\frac{1}{\rho_o}(\frac{1}{R}\frac{\partial qJ}{\partial \phi} + \frac{\partial qA_2}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial v}{\partial r})$$

$$\frac{\partial wJ}{\partial t} + \frac{1}{R\cos\phi}\frac{\partial uwJ}{\partial \lambda} + \frac{1}{R}\frac{\partial vwJ}{\partial \phi} + \frac{\partial w\omega}{\partial r} + \frac{u^2 + v^2}{R} = -\frac{1}{\rho_o}\frac{\partial q}{\partial r} + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial w}{\partial r}) + JF_w \quad (2.47)$$

$$\frac{1}{r\cos\phi}[\frac{\partial uJ}{\partial \lambda} + \frac{\partial vJ\cos\phi}{\partial \phi}] + \frac{\partial uA_1}{\partial r} + \frac{\partial vA_2}{\partial r} + \frac{\partial w}{\partial r} = 0 \quad (2.48)$$

$$\frac{\partial TJ}{\partial t} + \frac{1}{r\cos\phi}[\frac{\partial TuJ}{\partial \lambda} + \frac{\partial Tv\cos\phi J}{\partial \phi}] + \frac{\partial T\omega}{\partial r} = \frac{1}{J}\frac{\partial}{\partial r}(K_h\frac{\partial T}{\partial r}) + J\hat{H} + JF_T \quad (2.49)$$

$$\frac{\partial SJ}{\partial t} + \frac{1}{r\cos\phi}[\frac{\partial SuJ}{\partial \lambda} + \frac{\partial Sv\cos\phi J}{\partial \phi}] + \frac{\partial S\omega}{\partial r} = \frac{1}{J}\frac{\partial}{\partial r}(K_h\frac{\partial S}{\partial r}) + JF_S \quad (2.50)$$

$$\rho = \rho(\theta, S, p) \tag{2.51}$$

where u, v, and w are zonal, meridian and vertical components of the velocity, ω is the vertical velocity in the generalized terrain-following coordinate; T is the potential temperature; S is the salinity; ρ is the total density that is equal to a sum of perturbation density ρ' and reference density ρ_o, P is the pressure; f is the Coriolis parameter; g is the gravitational acceleration; K_m is the vertical eddy viscosity coefficient; and K_h is the thermal vertical eddy diffusion coefficient that are calculated using turbulence closure models (Chen *et al.*, 2004b). $\hat{H}$ is the vertical gradient of the short-wave radiation. F_u, F_v, F_T, and F_S represent the horizontal momentum, thermal, and salt diffusion terms and the horizontal diffusion is calculated using Smagorinsky's eddy parameterization method (Smagorinsky, 1963). $A_1 = \dfrac{J\partial r}{R\cos\phi\partial\lambda}$ and $A_2 = \dfrac{J\partial r}{R\partial\phi}$.

The 2-D (vertically integrated) momentum and continuity equations are written as

$$\frac{\partial\zeta}{\partial t} + \frac{1}{r\cos\varphi}[\frac{\partial\overline{u}D}{\partial\lambda} + \frac{\partial\overline{v}\cos\varphi D}{\partial\varphi}] = 0 \tag{2.52}$$

$$\frac{\partial\overline{u}}{\partial t} + \frac{1}{R\cos\phi}[\frac{\partial\overline{u}^2D}{\partial\lambda} + \frac{\partial\overline{u}\overline{v}D\cos\phi}{\partial\phi}] + \frac{\overline{u}\overline{v}D}{R}\tan\phi - \frac{\overline{wu}D}{R} - f\overline{v}D - D\tilde{F}_u - \frac{\tau_{s\lambda} - \tau_{b\lambda}}{\rho_o} - G_\lambda$$

$$= -\frac{gD}{R\cos\phi}\frac{\partial\zeta}{\partial\lambda} - \frac{D}{\rho_o R\cos\phi}\frac{\partial p_a}{\partial\lambda} - \frac{gD}{\rho_o R\cos\phi}[\int_{-1}^{0}\{J\int_{r}^{0}J(\frac{\partial\rho}{\partial\lambda} + \frac{\partial\rho}{\partial r'}\frac{\partial r}{\partial\lambda})dr']\}dr' \tag{2.53}$$

$$-\frac{1}{\rho_o R\cos\phi}\int_{-1}^{0}\frac{\partial qJ}{\partial\lambda}dr' - (qA_1\big|_{r=0} - qA_1\big|_{r=-1})$$

$$\frac{\partial\overline{v}}{\partial t} + \frac{1}{R\cos\phi}[\frac{\partial\overline{u}\overline{v}D}{\partial\lambda} + \frac{\partial\overline{v}^2D\cos\phi}{\partial\phi}] - \frac{\overline{u}^2D}{R}\tan\phi - \frac{\overline{wv}D}{R} + f\overline{u}D - D\tilde{F}_v - \frac{\tau_{s\phi} - \tau_{b\phi}}{\rho_o} - G_\phi$$

$$= -\frac{gD}{R}\frac{\partial\zeta}{\partial\phi} - \frac{D}{\rho_o R}\frac{\partial p_a}{\partial\phi} - \frac{gD}{\rho_o R}[\int_{-1}^{0}\{J\int_{r}^{0}J(\frac{\partial\rho}{\partial\phi} + \frac{\partial\rho}{\partial r'}\frac{\partial r}{\partial\phi})dr']\}dr' \tag{2.54}$$

$$-\frac{1}{\rho_o R}\int_{-1}^{0}\frac{\partial qJ}{\partial\phi}dr' - (qA_2\big|_{r=0} - qA_2\big|_{r=-1})$$

where G_u and G_v are defined as

$$G_u = \frac{1}{r\cos\phi}[\frac{\partial \overline{u}^2 D}{\partial \lambda} - \frac{\partial \overline{u^2} D}{\partial \lambda} + \frac{\partial \overline{u}\,\overline{v}\cos\phi D}{\partial \phi} - \frac{\partial \overline{uv}\cos\phi D}{\partial \phi}] + D\overline{F}_u - D\tilde{F}_u,$$
(2.55)

$$G_v = \frac{1}{r\cos\phi}[\frac{\partial \overline{u}\,\overline{v}D}{\partial \lambda} - \frac{\partial \overline{uv}D}{\partial \lambda} + \frac{\partial \overline{v}^2\cos\phi D}{\partial \phi} - \frac{\partial \overline{v^2}\cos\phi D}{\partial \phi}] + D\overline{F}_v - D\tilde{F}_v$$
(2.56)

and

$$D\tilde{F}_u \approx \frac{1}{r^2\cos^2\varphi}\frac{\partial}{\partial \lambda}[2\overline{A}_m H\frac{\partial \overline{u}}{\partial \lambda}] + \frac{1}{r}\frac{\partial}{\partial \varphi}[\overline{A}_m H(\frac{\partial \overline{u}}{r\partial \varphi} + \frac{\partial \overline{v}}{r\cos\varphi\partial \lambda})],$$
(2.57)

$$D\tilde{F}_v \approx \frac{\partial}{r\cos\phi\partial \lambda}[\overline{A}_m H(\frac{\partial \overline{u}}{r\partial \phi} + \frac{\partial \overline{v}}{r\cos\phi\partial})] + \frac{\partial}{r^2\partial \phi}[2\overline{A}_m H\frac{\partial \overline{v}}{\partial \phi}],$$
(2.58)

$$D\overline{F}_u \approx \frac{2}{r^2\cos^2\phi}\overline{\frac{\partial}{\partial \lambda}A_m H\frac{\partial u}{\partial \lambda}} + \frac{\partial}{r^2\partial \phi}\overline{A_m H(\frac{\partial u}{\partial \phi} + \frac{\partial v}{\cos\phi\partial \lambda})},$$
(2.59)

$$D\overline{F}_v \approx \frac{1}{r^2\cos\phi}\frac{\partial}{\partial \lambda}\overline{A_m H(\frac{\partial u}{\partial \phi} + \frac{1}{\cos\phi}\frac{\partial v}{\partial \lambda})} + \frac{2}{r^2}\frac{\partial}{\partial \phi}\overline{A_m H\frac{\partial v}{\partial \phi}}.$$
(2.60)

where the definitions of variables are the same as those described in the Cartesian coordinates. The spherical coordinate version of FVCOM is developed based on the Cartesian version, in which all the boundary conditions and forcings used in the spherical coordinate system are the same. The only difference is in the discrete approach, which is described in chapter 3.

2.5 The Turbulent Closure Models

2.5.1 The Horizontal Diffusion Coefficients. The primitive equations described above are not mathematically closed unless horizontal and vertical diffusion for momentum, temperature and salinity are determined. In FVCOM, users may choose between using a constant value for horizontal diffusion coefficient or the Smagorinsky eddy parameterization method (Smagorinsky, 1963). The Smagorinsky horizontal diffusion for momentum is given as

$$A_m = 0.5C\Omega^u\sqrt{(\frac{\partial u}{\partial x})^2 + 0.5(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial y})^2}$$
(2.61)

where C is a constant parameter and Ω^u is the area of the individual momentum control element (see Chapter 3 for definition). It is clear that the value of A_m varies with the

model resolution and the gradient of horizontal velocities: decreasing as the grid size or horizontal velocity gradients are reduced.

A similar formula is also used for scalars, which is proportional to the area of the individual tracer control element and the horizontal gradient of the tracer concentration. For water temperature, for example, it is given as

$$A_h = \frac{0.5C\Omega^\xi}{P_r} \sqrt{(\frac{\partial u}{\partial x})^2 + 0.5(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial y})^2} \tag{2.62}$$

where Ω^ξ is the area of the individual tracer control element (see Chapter 3 for definition) and P_r is the Prandtl number.

2.5.2 The Vertical Eddy Viscosity and Thermal Diffusion Coefficients. FVCOM features a wide choice of ocean turbulence closure models for the parameterization of the vertical eddy viscosity (K_m) and vertical thermal diffusion coefficient (K_h). The Mellor and Yamada (1982) level 2.5 (MY-2.5) turbulent closure model is the most popular $q - ql$ (where q is the turbulent kinetic energy and l is the turbulent macroscale) model. FVCOM features the updated version of the MY-2.5 model, which includes a) the upper and lower bounds limits of the stability function proposed by Galperin et al. (1988); b) the wind-driven surface wave breaking-induced turbulent energy input at the surface and internal wave parameterization by Mellor and Blumberg (2004); and c) the improved parameterization of pressure-strain covariance and shear instability-induced mixing in the strongly stratified region by Kantha and Clayson (1994, 2000).

Recently, the General Ocean Turbulent Model (GOTM) has become a very popular open-source community model (Burchard, 2002). GOTM implements a number of turbulence modules with a range from a simple Richardson parameterization to complex Reynolds-stress turbulence closure models. These modules include the MY-2.5 ($q - ql$) and ($k - \varepsilon$) turbulent closure models (where $k = q^1$ is the turbulent kinetic energy and ε is the turbulent dissipation). The $k - \varepsilon$ model is an alternative turbulent model that is very similar in dynamics to the $q - ql$ turbulent model. The most recent version of the $k - \varepsilon$ model also includes a more complete form of the pressure-strain covariance term with buoyancy, anisotropic production and vorticity contributions such that the cutoff of

---

[1] k is very popularly used in the European ocean modeling community.

mixing is shifted from $R_i = 0.2$ (original MY-2.5 model) to $R_i = 1.0$ (Canuto et al., 2001). The GOTM modules have been re-coded using the finite-volume approach to be consistent with the numerical methods used in FVCOM. Brief descriptions of the original MY-2.5 ($q - ql$) and the general form of the $k - \varepsilon$ model now featured in FVCOM are given below. Detailed descriptions of these models can be found in the GOTM manual and references listed in this paragraph.

2.5.3 The MY-2.5 Model. In the boundary layer approximation where the shear production of turbulent kinetic energy is produced by the vertical shear of the horizontal flow near the boundary, the equations for q^2 and $q^2 l$ can be simplified as

$$\frac{\partial q^2}{\partial t} + u\frac{\partial q^2}{\partial x} + v\frac{\partial q^2}{\partial y} + w\frac{\partial q^2}{\partial z} = 2(P_s + P_b - \varepsilon) + \frac{\partial}{\partial z}(K_q \frac{\partial q^2}{\partial z}) + F_q \qquad (2.63)$$

$$\frac{\partial q^2 l}{\partial t} + u\frac{\partial q^2 l}{\partial x} + v\frac{\partial q^2 l}{\partial y} + w\frac{\partial q^2 l}{\partial z} = lE_1(P_s + P_b - \frac{\tilde{W}}{E_1}\varepsilon) + \frac{\partial}{\partial z}(K_q \frac{\partial q^2 l}{\partial z}) + F_l \qquad (2.64)$$

where $q^2 = (u'^2 + v'^2)/2$ is the turbulent kinetic energy; l is the turbulent macroscale; K_q is the vertical eddy diffusion coefficient of the turbulent kinetic energy; F_q and F_l represent the horizontal diffusion of the turbulent kinetic energy and macroscale; $P_s = K_m(u_z^2 + v_z^2)$ and $P_b = (gK_h\rho_z)/\rho_o$ are the shear and buoyancy production terms of turbulent kinetic energy; $\varepsilon = q^3/B_1 l$ is the turbulent kinetic energy dissipation rate; $W = 1 + E_2 l^2/(\kappa L)^2$ is a wall proximity function where $L^{-1} = (æ - z)^{-1} + (H + z)^{-1}$; $\kappa = 0.4$ is the von Karman constant; H is the mean water depth; and æ is the free surface elevation. In general, F_q and F_l are kept as small as possible to reduce the effects of horizontal diffusion on the solutions. In FVCOM, F_q and F_l are parameterized using the Smagorinsky formulation shown above. However, the turbulent closure model can be run with both F_q and F_l set to zero.

The turbulent kinetic energy and macroscale equations are closed by defining

$$K_m = lqS_m, \ K_h = lqS_h, \ K_q = 0.2lq \qquad (2.65)$$

S_m and S_h are defined as the stability functions

$$S_m = \frac{0.4275 - 3.354 G_h}{(1 - 34.676 G_h)(1 - 6.127 G_h)} \text{ and } S_h = \frac{0.494}{1 - 34.676 G_h} \tag{2.66}$$

where $G_h = \frac{l^2 g}{q^2 \rho_o} \rho_z$. In the original MY-2.5 turbulent closure model (Mellor and

Yamada, 1974, 1982), S_m and S_h are functions of the gradient Richardson number. By

removing a slight inconsistency in the scaling analysis, Galperin et al. (1988) simplified

the MY turbulent closure model so that S_m and S_h depend only on G_h. G_h has an upper

bound of 0.023 for the case of unstable ($\rho_z > 0$) stratification and a lower bound of -0.28

for the case of stable ($\rho_z < 0$) stratification. Parameters A_1, A_2, B_1, B_2, and C_1 are given

as 0.92, 16.6, 0.74, 10.1, and 0.08, respectively.

In the original MY-2.5 model, the surface and bottom boundary conditions for the

turbulent kinetic energy and macroscale equations are given as

$$q^2 l = 0, \quad q^2 = B_1^{\frac{2}{3}} u_{\tau s}^2, \text{ at } z = \zeta (x, y, t), \tag{2.67}$$

$$q^2 l = 0, \quad q^2 = B_1^{\frac{2}{3}} u_{\tau b}^2, \text{ at } z = -H(x,y), \tag{2.68}$$

where $u_{\tau s}$ and $u_{\tau b}$ are the water friction velocities associated with the surface and bottom.

Since $q^2 \neq 0$ at the surface and bottom, l equals zero at the boundaries. This means that

K_m, K_h and K_q always equal zero at the surface and bottom. This simplification is

reasonable for the bottom but ignores the turbulent energy flux due to surface waves

during windy conditions.

Mellor and Blumberg (2004) introduced a new turbulent kinetic flux surface

boundary condition into the MY- 2.5 model, in which

$$\frac{\partial q^2}{\partial z} = \frac{2 \alpha_{CB} u_{\tau s}^3}{K_q}; \ l = \max(k z_w, l_z) \text{ at } z = \zeta (x, y, t) \tag{2.69}$$

where α_{CB} is a parameter related to the wave age; l_z is the "conventional" empirical

length scale; $\kappa = 0.4$ is the von Karman constant and z_w is the wave-related roughness

height. According to the best fit to available observational data (Terray et al., 1996, 1997),

α_{CB} can be approximated by

$$\alpha_{CB} = 15 \frac{c_p}{u_*} \exp[-(0.04 c_p / u_*)^4] \tag{2.70}$$

where c_p is the phase speed of wave at the dominant frequency, u_* is the air friction velocity ($u_* = 30 u_\tau$), and c_p / u_* is the "wave age". The value of α_{CB} changes significantly with the wave age: it is given as

$$\alpha_{CB} \cong \begin{cases} 0 & \text{for } c_p / u_* = 0 \quad \text{no waves : original MY 2.5 model} \\ 146 & \text{for } c_p / u_* = 10 \quad \text{younger waves} \\ 57 & \text{for } c_p / u_* = 30 \quad \text{mature waves} \end{cases} \tag{2.71}$$

In general, l_z is proportional to z, which can be approximately estimated by

$$l_z = \kappa z \tag{2.72}$$

According to a better fit to available observational data (Terray et al., 2000; Mellor and Blumberg, 2004), z_w can be determined by

$$z_w = 0.85 H_s \tag{2.73}$$

where H_s is the significant wave height defined as $4 H_{rms}$ (H_{rms} is the rms wave height). As suggested by Donelan (1990) and Smith et al. (1992), H_s can be estimated based on the wave age and airside roughness parameter (z_o) in a form of

$$H_s = 2.0 (\frac{c_p}{u_*})^{2.5} z_o \tag{2.74}$$

Specifying $z_o = \alpha_{CH} u_* / g$ (Charnock's relation), $\alpha_{CH} = 0.45 u_* / c_p$ (Smith et al., 1992 Janssen, 2001) and $u_* = (\rho_w / \rho_a) u_\tau^2$, z_w can be rewritten as

$$z_w = \beta \frac{u_\tau^2}{g}; \quad \beta = 665 \left(\frac{c_p}{u_*} \right)^{1.5} . \tag{2.75}$$

According to observational data, $\beta = 2.0 \times 10^5$ (Stacey, 1999).

2.5.4. The $k - \varepsilon$ Turbulent Model. In the boundary layer approximation (Rodi, 1980), the $k - \varepsilon$ model can be simplified as

$$\frac{\partial k}{\partial t} - \frac{\partial}{\partial z} (\frac{\nu_t}{\hat{\sigma}_k} \frac{\partial k}{\partial z}) = P + G - \varepsilon \tag{2.76}$$

$$\frac{\partial \varepsilon}{\partial t} - \frac{\partial}{\partial z}(\frac{v_t}{\hat{\sigma}_\varepsilon}\frac{\partial \varepsilon}{\partial z}) = c_1(P + c_3 G)\frac{\varepsilon}{k} - c_2\frac{\varepsilon^2}{k} \tag{2.77}$$

where v_t is the eddy viscosity (which is the same as K_q in the MY-2.5 model), $\hat{\sigma}_k$ is the turbulent Prandtl number that is defined as the ratio of turbulent eddy viscosity to the thermal diffusivity, P is the turbulent shear production, and G is the turbulent buoyancy production. These two variables have the same definitions as P_s and P_b in the MY-2.5 model. c_1, c_2 and c_3 are empirical constants. A detailed description of the standard and advanced $k - \varepsilon$ models was given by Burchard and Baumert (1995) and is briefly summarized next.

In the standard $k - \varepsilon$ model,

$$P = -\overline{u'w'}\frac{\partial \overline{u}}{\partial z} - \overline{v'w'}\frac{\partial \overline{v}}{\partial z} = v_t[(\frac{\partial \overline{u}}{\partial z})^2 + (\frac{\partial \overline{v}}{\partial z})^2] \tag{2.78}$$

$$G = -\frac{g}{\rho_o}\overline{w'\rho'} = -\frac{g}{\rho_o}(\frac{v_t}{\hat{\sigma}_k})\frac{\partial \overline{\rho}}{\partial z} \tag{2.79}$$

where

$$\hat{\sigma}_k = \begin{cases} \dfrac{[1 + (10/3)R_i]^{3/2}}{(1 + 10R_i)^{1/2}} & R_i \geq 0 \\ 1 & R_i < 0 \end{cases} \tag{2.80}$$

and R_i is the gradient Richardson number defined as

$$R_i = \frac{N_G^2}{N_P^2}; \ N_G^2 = -\frac{g}{\rho_o}\frac{\partial \overline{\rho}}{\partial z}; \ N_P^2 = (\frac{\partial \overline{u}}{\partial z})^2 + (\frac{\partial \overline{v}}{\partial z})^2 \tag{2.81}$$

The eddy viscosity v_t can be estimated by

$$v_t = c_\mu \frac{k^2}{\varepsilon} \tag{2.82}$$

where c_μ is a constant. In this standard $k - \varepsilon$ model, the empirical constants are specified as

$$(c_\mu, c_1, c_2, \hat{\sigma}_k, \hat{\sigma}_\varepsilon) = (0.09, 1.44, 1.92, 1.00, 1.30) \tag{2.83}$$

In the advanced $k - \varepsilon$ model, the turbulence model consists of the k and ε equations plus 6 transport equations for the Reynolds stresses ($\overline{u'w'}$, $\overline{v'w'}$ and $\overline{w'^2}$) and the

turbulent heat fluxes ($\overline{u'T'}$, $\overline{v'T'}$ and $\overline{w'T'}$). In this model, the eddy viscosity (v_t) is still given by (2.59), but c_μ is a function of the vertical shear of the horizontal velocity and vertical stratification. This function corresponds to the stability function S_m in the MY-2.5 model. v_t and v_T (thermal diffusion coefficient) are given as

$$v_t = c_\mu(\alpha_P, \alpha_{G,}F)\frac{k^2}{\varepsilon}, \quad v_t = c'_\mu(\alpha_P, \alpha_{G,}F)\frac{k^2}{\varepsilon} \tag{2.84}$$

where α_P and α_G are functions of dimensionless turbulent shear and turbulent buoyancy numbers in the forms of

$$\alpha_P = \frac{k^2}{\varepsilon^2}N_P^2; \quad \alpha_G = \frac{k^2}{\varepsilon^2}N_G^2 . \tag{2.85}$$

F is a near-wall correction factor.

The 8-component advanced turbulence model is mathematically closed with the specification of 11 empirical constants (Burchard and Baumert, 1995).

The surface boundary conditions for k and ε in the k-ε turbulent model described above are specified as

$$v_t \frac{\partial k}{\partial z} = 0, \qquad \text{if } kc_\mu^{-1/2} > u_{\tau s}^2$$

$$k = u_{\tau s}^2 / c_\mu^{1/2}, \quad \text{otherwise} \tag{2.86}$$

$$\varepsilon = \frac{k^{3/2}c_\mu^{3/4}}{\kappa\{H + z + 0.07H[1 - (u_{\tau s}^2 / kc_\mu^{1/2})]\}} .$$

The bottom boundary conditions for k and ε are given as

$$k = u_{\tau b}^2 / c_\mu^{1/2}$$

$$\varepsilon = \frac{1}{\kappa(H + z)}u_{\tau b}^3 \tag{2.87}$$

where κ is the von Karman constant.

The wave-induced turbulent kinetic energy flux at the surface was recently taken into account for the $k - \varepsilon$ model. A detailed description of the modified surface boundary conditions for k and ε is given in Burchard (2001).

Chapter 3: The Finite-Volume Discrete Method

The original version of FVCOM was developed in the σ-coordinate transformation system. The code was subsequently upgraded to the generalized terrain-following coordinate system in 2006. The discretization forms of the governing equations have been significantly modified in this new coordinate system. When the non-hydrostatic version of FVCOM was developed in 2008, we implemented a semi-implicit solver, so that the current version of FVCOM has two options for the time integration: 1) mode-split and 2) semi-implicit. In this chapter, we provide an example of the discrete forms of FVCOM in the σ-coordinate transformation system for the mode-split solver. The σ-coordinate transformation is one selection of the generalized terrain-following coordinates, so learning the details of the discretization forms in this coordinate system can help users learn how the generalized terrain-following coordinates work in FVCOM. A brief description of the semi-implicit solver is given in Chapter 4 when the non-hydrostatic solver is introduced. Users who are interested to learning the details of discretization forms in the generalized terrain-following coordinate system can examine directly from the source code directly.

3.1. Design of the Unstructured Triangular Grids
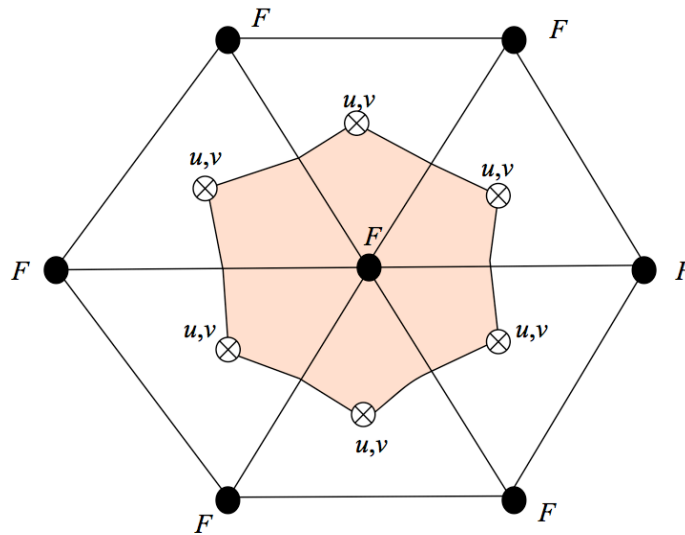
Fig. 3.1: Illustration of the FVCOM unstructured triangular grid. Variable locations: Node •: H, ζ, ω, D, s, θ, q^2, $q^2 l$, A_m, K_h; Centroid ⊗: u, v. F represents all tracer variables.

Similar to a triangular finite element method, the horizontal numerical computational domain is subdivided into a set of non-overlapping unstructured triangular cells. An unstructured triangle is comprised of three nodes, a centroid, and three sides (Fig. 3.1). Let N and M be the total number of centroids and nodes in the computational domain, respectively, then the locations of centroids can be expressed as:

$$[X(i), Y(i)], i = 1:N,$$ (3.1)

and the locations of nodes can be specified as:

$$[X_n(j), Y_n(j)], j = 1:M.$$ (3.2)

Since none of the triangles in the grid overlap, N should also be the total number of triangles. On each triangular cell, the three nodes are identified using integral numbers defined as $N_i(\hat{j})$ where $\hat{j}$ is counted clockwise from 1 to 3. The surrounding triangles that have a common side are counted using integral numbers defined as $NBE_i(\hat{j})$ where $\hat{j}$ is counted clockwise from 1 to 3. At open or coastal solid boundaries, $NBE_i(\hat{j})$ is specified as zero. At each node, the total number of the surrounding triangles with a connection to this node (for example, the jth node) is expressed as $NT(j)$, and they are counted using integral numbers $NB_i(m)$ where m is counted clockwise from 1 to $NT(j)$.

To provide a more accurate estimation of the sea-surface elevation, currents and salt and temperature fluxes, u and v are placed at centroids and all scalar variables, such as ζ, H, D, ω, S, T,

Fig. 3.2: The location of the model variables in the vertical sigma coordinate.

ρ, K_m, K_h, A_m and A_h are placed at nodes. Scalar variables at each node are determined by a net flux through the sections linked to centroids and the mid-point of the adjacent sides

in the surrounding triangles (called the "tracer control element" or TCE), while u and v at the centroids are calculated based on a net flux through the three sides of that triangle (called the "momentum control element" or MCE).

Similar to finite-difference models such as POM and ROMs, all the model variables except ω (vertical velocity on the sigma-layer surface) and turbulence variables (such as q^2 and $q^2 l$) are placed at the mid-level of each σ layer (Fig. 3.2). There are no restrictions on the thickness of the σ-layer, which allows users to use either uniform or non-uniform σ-layers.

3.2. The Discrete Procedure in Cartesian Coordinates

3.2.1. The 2-D External Mode. Let us consider the continuity equation first. Integrating Eq. (2.30) over a given triangle area yields:

$$\iint \frac{\partial \zeta}{\partial t} dxdy = -\iint [\frac{\partial (\bar{u}D)}{\partial x} + \frac{\partial (\bar{v}D)}{\partial x}]dxdy = -\oint_{s'} \bar{v}_n Dds', \tag{3.3}$$

where $\bar{v}_n$ is the velocity component normal to the sides of the triangle and s' is the closed trajectory comprised of the three sides. Eq. (3.3) is integrated numerically using the modified fourth-order Runge-Kutta time-stepping scheme. This is a multi-stage time-stepping approach with second-order temporal accuracy. The detailed procedure for this method is described as follows:

$$\zeta_j^0 = \zeta_j^n, \ R_\zeta^0 = R_\zeta^n = \sum_{m=1}^{NT(j)} [(\Delta x_{2m-1}\bar{v}_m^n - \Delta y_{2m-1}\bar{u}_m^n)D_{2m-1}^n + (\Delta x_{2m}\bar{v}_m^n - \Delta y_{2m}\bar{u}_m^n)D_{2m}^n], \tag{3.4}$$

$$\zeta_j^k = \zeta_j^0 - \alpha^k \frac{\Delta t R_\zeta^{k-1}}{2\Omega_j^\xi}; \text{ and } \zeta_j^{n+1} = \zeta_j^4, \tag{3.5}$$

where $k = 1,2,3,4$ and $(\alpha^1, \alpha^2, \alpha^3, \alpha^4) = (1/4, 1/3, 1/2, 1)$. Superscript n represents the nth time step. Ω_j^ξ is the area enclosed by the lines through centroids and mid-points of the sides of surrounding triangles connected to the node where ζ_j is located. $\bar{u}_m^n$ and $\bar{v}_m^n$ are defined as:

$$\bar{u}_m^n = \overline{u(NT(m))}^n, \bar{v}_m^n = \overline{v(NT(m))}^n. \tag{3.6}$$

Δt is the time step for the external mode, and

$$\Delta x_{2m-1} = x_{2m} - x_{2m-1}; \Delta x_{2m} = x_{2m+1} - x_{2m}; \tag{3.7}$$

$$\Delta y_{2m-1} = y_{2m} - y_{2m-1}; \Delta y_{2m} = y_{2m+1} - y_{2m}. \tag{3.8}$$

Similarly, integrating Eqs. (2.31) and (2.32) over a given triangular area gives:

$$\iint \frac{\partial \bar{u} D}{\partial t} dx dy = -\oint_{s'} \bar{u} D \bar{v}_n ds' + \iint f \bar{v} D dx dy - \iint g D \frac{\partial \xi}{\partial x} dx dy$$

$$- \iint \{\frac{g D^2}{\rho_o} \int_{-1}^{0} [\frac{\partial}{\partial x} \int_{\sigma}^{0} \rho d\sigma - \int_{\sigma}^{0} \frac{\partial \rho}{\partial \sigma} \frac{\sigma}{D} \frac{\partial D}{\partial x} d\sigma] d\sigma\} dx dy$$

$$+ \iint \frac{\tau_{sx} - \tau_{bx}}{\rho_o} dx dy + \iint D \widetilde{F}_x dx dy + \iint G_x dx dy \tag{3.9}$$

$$\iint \frac{\partial \bar{v} D}{\partial t} dx dy = -\oint_{s'} \bar{v} D \bar{v}_n ds' - \iint f \bar{u} D dx dy - \iint g D \frac{\partial \xi}{\partial y} dx dy$$

$$- \iint \{\frac{g D^2}{\rho_o} \int_{-1}^{0} [\frac{\partial}{\partial y} \int_{\sigma}^{0} \rho d\sigma - \int_{\sigma}^{0} \frac{\partial \rho}{\partial \sigma} \frac{\sigma}{D} \frac{\partial D}{\partial y} d\sigma] d\sigma\} dx dy$$

$$+ \iint \frac{\tau_{sy} - \tau_{by}}{\rho_o} dx dy + \iint D \widetilde{F}_y dx dy + \iint G_y dx dy. \tag{3.10}$$

Eqs. (3.9) and (3.10) are also integrated numerically using the modified fourth-order Runge-Kutta time-stepping scheme as follows:

$$\bar{u}_i^0 = \bar{u}_i^n, \bar{v}_i^0 = \bar{v}_i^n; \overline{R}_u^0 = \overline{R}_u^n, \overline{R}_v^0 = \overline{R}_v^n, \tag{3.11}$$

$$\bar{u}_i^k = \bar{u}_i^0 - \alpha^k \frac{\Delta t \overline{R}_u^0}{4 \Omega_i^u \overline{D}_i}, \bar{v}_i^k = \bar{v}_i^0 - \alpha^k \frac{\Delta t \overline{R}_v^0}{4 \Omega_i^v \overline{D}_i}, \tag{3.12}$$

$$\bar{u}_i^{n+1} = \bar{u}_i^4, \bar{v}_i^{n+1} = \bar{v}_i^4 \tag{3.13}$$

where the definitions of k and α^k are the same as those shown in Eqs. (3.4) and (3.5). Ω_i^u and Ω_i^v are the triangular areas where $\bar{u}$ and $\bar{v}$ are located. In the grids used in this model, $\bar{u}$ and $\bar{v}$ are always located at the centroid, so that $\Omega_i^u = \Omega_i^v = \Omega_i$. $\overline{D}_i$ is the depth at the centroid, which is interpolated from depth values at the three surrounding nodes. $\overline{R}_u^n$ and $\overline{R}_v^n$ represent all the terms on the right of Eqs. (3.9) and (3.10), respectively. They are equal to

$$\overline{R}_u^n = ADVU + DPBPX + DPBCX + CORX + VISCX - G_x, \tag{3.14}$$

$$\overline{R}_v^n = ADVV + DPBPY + DPBCY + CORY + VISCY - G_y, \tag{3.15}$$

where *ADVU* and *ADVV*, *DPBPX* and *DPBPY*, *DPBCX* and *DPBCY*, *CORX* and *CORY*, *VISCX* and *VISCY* are the *x* and *y* components of the vertically integrated horizontal advection, barotropic pressure gradient force, Coriolis force, and horizontal diffusion terms, respectively. The definitions of G_x and G_y are the same as those shown in Eqs. (2.33) and (2.34) in the text.

The *x* and *y* components of the horizontal advection are calculated numerically by

$$ADVU = \sum_{m=1}^{3} \left(\bar{u}_{im} \overline{D}_m * \bar{v}_{nm} \hat{l}_m \right), \ ADVV = \sum_{m=1}^{3} \left(\bar{v}_{im} \overline{D}_m * \bar{v}_{nm} \hat{l}_m \right), \tag{3.16}$$

where $\bar{u}_{im}$, $\bar{v}_{im}$, and $\bar{v}_{nm}$ are the *x*, *y* and normal components of the velocity on the edge *m* of a triangle cell, and $\bar{v}_{nm}$ is positive when its direction is outward. $\hat{l}_m$ and $\overline{D}_m$ are the length and mid-point water depth of the side *m*, respectively. They are equal to

$$\overline{D}_m = 0.5[D(N_i(j_1)) + D(N_i(j_2))] , \tag{3.17}$$

$$\hat{l}_m = \sqrt{[X_n(N_i(j_1)) - X_n(N_i(j_2))]^2 + [Y_n(N_i(j_1)) - Y_n(N_i(j_2))]^2} , \tag{3.18}$$

where

$$j_2 = m + 1 - \text{INT}(\frac{m+1}{4}) \times 3; \ j_1 = m + 2 - \text{INT}(\frac{m+2}{4}) \times 3. \tag{3.19}$$

The velocity in the triangle cell *i* is assumed to satisfy the linear distribution given as
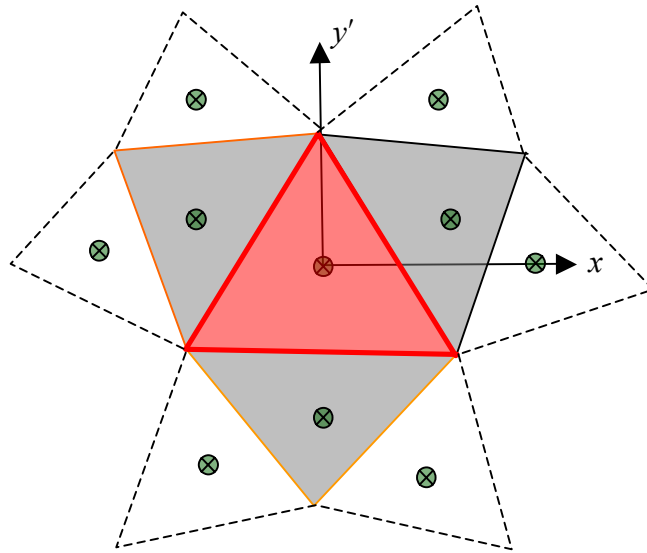
Fig. 3.3: Illustration of the local coordinate used to calculate the velocity and triangular cells used to determine the linear function of the horizontal velocity.

$$\overline{u}_i(x',y') = \phi_i^u(x',y') = \overline{u}_{i,0} + a_i^u x' + b_i^u y' \; , \tag{3.20}$$

$$\overline{v}_i(x',y') = \phi_i^v(x',y') = \overline{v}_{i,0} + a_i^v x' + b_i^v y' \; , \tag{3.21}$$

where the parameters a_i^u, b_i^u, a_i^v, and b_i^v are determined by a least-square method based on velocity values at the four cell centered points shown in Fig. 3.3 (one calculated cell (shaded red) plus three surrounding cells). Then, the normal velocity component on the side m is given as

$$\overline{v}_{nm} = \widehat{v}_m \cos\widehat{\theta} - \widehat{u}_m \sin\widehat{\theta}, \tag{3.22}$$

where

$$\widehat{\theta} = \arctan\frac{Y_n(N_i(j_2)) - Y_n(N_i(j_1))}{X_n(N_i(j_2)) - X_n(N_i(j_i))} \tag{3.33}$$

and

$$\widehat{u}_{im} = 0.5[\phi_i^u(\overline{x}_m', \overline{y}_m') + \phi_{NBi(m)}^u(\overline{x}_m', \overline{y}_m')], \; \widehat{v}_{im} = 0.5[\phi_i^v(\overline{x}_m', \overline{y}_m') + \phi_{NBi(m)}^v(\overline{x}_m', \overline{y}_m')], \tag{3.34}$$

where $\overline{x}_m'$ and $\overline{y}_m'$ are the mid-point of the side.

The momentum flux through the three sides of triangle cell i is calculated using a second-order accurate scheme (Kobayashi, 1999) as follows:

$$\overline{u}_{im} = \begin{cases} \phi_i^u(0,0), & \overline{v}_{nm} < 0 \\ \phi_{NB_i(m)}^u(x_{im}, y_{im}), & \overline{v}_{nm} \geq 0 \end{cases}, \; \overline{v}_{im} = \begin{cases} \phi_i^v(0,0), & \overline{v}_{nm} < 0 \\ \phi_{NB_i(m)}^v(x_{im}, y_{im}), & \overline{v}_{nm} \geq 0 \end{cases} \tag{3.35}$$

where x_{im} and y_{im} are the cell-centered point of the surrounding triangle numbered $NB_i(m)$, and $(0,0)$ indicates the location of the cell-centered point.

In the updated code, instead of calculating the normal velocity for each component on the side m, we directly calculate the flux through the side m, which is equal to

$$\overline{V}_{nm} = \widehat{v}_n(X_n(N_i(j_2)) - X_m(N_i(j_1)) - \widehat{u}_m(Y_n(N_i(j_2)) - Y_n(N_i(j_1))) \; . \tag{3.36}$$

This method significantly improves the numerical accuracy by removing the calculation of the angle $\widehat{\theta}$.

The area integration of the barotropic pressure gradient force terms can be converted to a trajectory integration using Stokes' theorem. They can then be calculated numerically by a simple discrete method as follows:

$$DPBPX = g\bar{D}_i \sum_{m=1}^{3} \bar{\zeta}_m [Y_n(N_i(j_1)) - Y_n(N_i(j_2))], \tag{3.37}$$

$$DPBPY = g\bar{D}_i \sum_{m=1}^{3} \bar{\zeta}_m [X_n(N_i(j_2)) - X_n(N_i(j_1))], \tag{3.38}$$

where $\bar{\zeta}_m = 0.5[\zeta(N_i(j_1)) + \zeta(N_i(j_2))]$.

A similar approach is used to calculate the baroclinic pressure gradient force terms. These terms are rewritten into the form of the gradient to take advantage of the flux calculation in the finite-volume method. For example, the x component of the baroclinic pressure gradient force can be rewritten as:

$$
\begin{aligned}
&-\frac{gD}{\rho_o} \int_{\sigma}^{0} (D\frac{\partial \rho'}{\partial x} - \sigma\frac{\partial D}{\partial x}\frac{\partial \rho'}{\partial \sigma})d\sigma \\
&= -\frac{gD}{\rho_o} [\int_{\sigma}^{0} (D\frac{\partial \rho'}{\partial x} + \rho\frac{\partial D}{\partial x} - \rho\frac{\partial D}{\partial x} - \sigma\frac{\partial D}{\partial x}\frac{\partial \rho'}{\partial \sigma})d\sigma] \\
&= -\frac{gD}{\rho_o} \int_{\sigma}^{0} [\frac{\partial D\rho'}{\partial x} - \frac{\partial D}{\partial x}(\rho + \sigma\frac{\partial \rho'}{\partial \sigma})]d\sigma \\
&= -\frac{gD}{\rho_o} [\int_{\sigma}^{0} \frac{\partial D\rho'}{\partial x}d\sigma - \frac{\partial D}{\partial x}(\int_{\sigma}^{0} \rho d\sigma + \int_{\sigma}^{0} \sigma\frac{\partial \rho'}{\partial \sigma}d\sigma)] \\
&= -\frac{gD}{\rho_o} [\int_{\sigma}^{0} \frac{\partial D\rho'}{\partial x}d\sigma - \frac{\partial D}{\partial x}(-\sigma\rho - \int_{\sigma}^{0} \sigma\frac{\partial \rho'}{\partial \sigma}d\sigma + \int_{\sigma}^{0} \sigma\frac{\partial \rho'}{\partial \sigma}d\sigma)] \\
&= -\frac{gD}{\rho_o} [\int_{\sigma}^{0} \frac{\partial D\rho'}{\partial x}d\sigma + \sigma\rho\frac{\partial D}{\partial x}] \\
&= -\frac{gD}{\rho_o} \{\frac{\partial}{\partial x}[D\int_{\sigma}^{0} \rho d\sigma + \sigma\rho D] - D\frac{\partial \rho\sigma}{\partial x}\} \\
&= -\frac{gD}{\rho_o} [\frac{\partial}{\partial x}(D\int_{\sigma}^{0} \sigma\frac{\partial \rho'}{\partial \sigma}d\sigma) - D\frac{\partial \rho\sigma}{\partial x}]
\end{aligned}
\tag{3.39}
$$

Integrating Eq. (3.39) from -1 to 0 and then integrating over a triangle cell area again, we get

$$
\begin{aligned}
DPBCX &= \frac{g}{\rho_o} \{\iint [D\frac{\partial}{\partial x}\int_{-1}^{0}(D\int_{\sigma}^{0} \rho\frac{\partial \rho}{\partial \sigma'}d\sigma')d\sigma]dxdy + \iint D^2\frac{\partial}{\partial x}(\int_{-1}^{0} \rho\sigma d\sigma)dxdy\} \\
&= \frac{g}{\rho_o} \{\bar{D}\oint[D\int_{-1}^{0}(\int_{\sigma}^{0} \sigma\frac{\partial \rho}{\partial \sigma'}d\sigma')d\sigma]dy + \bar{D}^2\oint(\int_{-1}^{0} \rho\sigma d\sigma)dy\}
\end{aligned}
\tag{3.40}
$$

The discrete form of Eq. (3.40) is given as

$$DPBCX = \frac{0.5g}{\rho_0}\{\overline{D}_i \sum_{m=1}^{3}\overline{D}_m[PB_1(i) + PB_2(NB_i(m)][Y_n(N_i(j_1)) - Y_n(N_i(j_2))]$$

$$+\overline{D}_i^2 \sum_{m=`1}^{3}[PB_2(i) + PB_2(NB_i(m))][Y_n(N_i(j_1)) - Y_n(N_i(j_2))]\} \tag{3.41}$$

where

$$PB_1(i) = \sum_{k'=1}^{KB-1}\{[\sigma(k') - \sigma(k'+1)]\sum_{k=1}^{k'}\sigma(k)[\rho(k) - \rho(k+1)]\}, \tag{3.42}$$

$$PB_2(i) = 0.5\sum_{k'=1}^{KB-1}[\rho(k) + \rho(k+1)]\sigma(k)[\sigma(k) - \sigma(k+1)]. \tag{3.43}$$

Similarly, we can derive the y component of the baroclinic pressure gradient force as

$$DPBCY = \frac{0.5g}{\rho_0}\{\overline{D}_i \sum_{m=1}^{3}\overline{D}_m[PB_1(i) + PB_2(NB_i(m)][X_n(N_i(j_2)) - X_n(N_i(j_1))]$$

$$+\overline{D}_i^2 \sum_{m=`1}^{3}[PB_2(i) + PB_2(NB_i(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))]\}. \tag{3.44}$$

The discrete forms of the Coriolis force terms are given as

$$CORX = -fv_i D_i \Omega_i^u; \quad CORY = fu_i D_i \Omega_i^v \tag{3.45}$$

In the updated FVCOM code, we have implemented a semi-implicit scheme to calculate the Coriolis force term with a weight-averaged velocity at time step *n* and *n+1*.

The *x* and *y* components of the horizontal diffusion can be rewritten as

$$\iint D\widetilde{F}_x dxdy \approx \iint\{\frac{\partial}{\partial x}(2\overline{A}_m H\frac{\partial\overline{u}}{\partial x}) + \frac{\partial}{\partial y}[\overline{A}_m H(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x})]\}dxdy$$

$$= 2\oint\overline{A}_m H\frac{\partial\overline{u}}{\partial x}dy - \oint\overline{A}_m H(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x})dx \tag{3.46}$$

and

$$\iint D\widetilde{F}_y dxdy \approx \iint\{\frac{\partial}{\partial y}(2\overline{A}_m H\frac{\partial\overline{v}}{\partial y}) + \frac{\partial}{\partial x}[\overline{A}_m H(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x})]\}dxdy$$

$$= -2\oint\overline{A}_m H\frac{\partial\overline{v}}{\partial y}dx + \oint\overline{A}_m H(\frac{\partial\overline{u}}{\partial y} + \frac{\partial\overline{v}}{\partial x})dy \tag{3.47}$$

The discrete forms of Eqs. (3.46) and (3.47) are given as

$$VISCX = \sum_{m=1}^{3} \{0.5\overline{H}_m[\overline{A}_m(i) + \overline{A}_m(NB(m))][a^u(i) + a^u(NB(m))]$$

$$[Y_n(N_i(j_1)) - Y_n(N_i(j_2))] + 0.25\overline{H}_m[\overline{A}_m(i) + \overline{A}_m(NB(m))] \tag{3.48}$$

$$[[b^u(i) + b^u(NB(m)) + a^v(i) + a^v(NB(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))]\}$$

where $\overline{H}_m = 0.5[H(N_i(j_1) + H(N_i(j_2))]$, and

$$VISCY = \sum_{m=1}^{3} \{0.5\overline{H}_m[\overline{A}_m(i) + \overline{A}_m(NB(m))][b^v(i) + b^v(NB(m))]$$

$$[X_n(N_i(j_2)) - X_n(N_i(j_1))] + 0.25\overline{H}_m[\overline{A}_m(i) + \overline{A}_m(NB(m))] \tag{3.49}$$

$$[[b^u(i) + b^u(NB(m)) + a^v(i) + a^v(NB(m))][Y_n(N_i(j_1)) - Y_n(N_i(j_2))]\}.$$

The G_x and G_y terms are given as

$$G_x = ADVU + VICX - \overline{ADVU} - \overline{VISCX} , \tag{3.50}$$

$$G_y = ADVV + VICY - \overline{ADVV} - \overline{VISCY} , \tag{3.51}$$

where

$$\overline{ADVU} = \iint[\frac{\partial \overline{u^2}D}{\partial x} + \frac{\partial \overline{uv}D}{\partial y}]dxdy = \oint \overline{u^2}Ddy + \oint \overline{uv}Ddx$$

$$= \sum_{m=1}^{3} 0.5\{[\overline{u^2(i)} + \overline{u^2(NB(m))}]\overline{D}_m[Y_n(N_i(j_1)) - Y_n(N_i(j_2))] \tag{3.52}$$

$$+ [\overline{u(i)v(i)} + \overline{u(NB(m))v(NB(m))}]\overline{D}_m[X_n(N_i(j_2)) - X_n(N_i(j_1))]\};$$

$$\overline{ADVV} = -\iint[\frac{\partial \overline{uv}D}{\partial x} + \frac{\partial \overline{v^2}D}{\partial y}]dxdy = -\oint \overline{uv}Ddy - \oint \overline{v^2}Ddx$$

$$= \sum_{m=1}^{3} 0.5\{[\overline{u(i)v(i)} + \overline{u(NB(m))v(NB(m))}]\overline{D}_m[Y_n(N_i(j_1)) - Y_n(N_i(j_2))] \tag{3.53}$$

$$+ [\overline{v^2(i)} + \overline{v^2(NB(m))}]\overline{D}_m[X_n(N_i(j_2)) - X_n(N_i(j_1))]\};$$

$$\overline{VISCX} = \iint D\overline{F}_x dxdy \approx \iint[\frac{\partial}{\partial x}\overline{2A_mH\frac{\partial u}{\partial x}} + \frac{\partial}{\partial y}\overline{A_mH(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]dxdy$$

$$= \oint\overline{(2A_mH\frac{\partial u}{\partial x})}dy - \oint\overline{[A_mH(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})]}dx \tag{3.54}$$

$$= 2\oint H\overline{(A_m\frac{\partial u}{\partial x})}dy - \oint H\overline{[A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})]}dx;$$

and

$$\overline{VISCY} = \iint D\overline{F}_y \, dxdy \approx \iint [\frac{\partial}{\partial y}\overline{2A_m H \frac{\partial v}{\partial y}} + \frac{\partial}{\partial x}\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]dxdy$$

$$= -\oint \overline{(2A_m H \frac{\partial v}{\partial y})}\, dx + \oint [\overline{A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]\, dy \qquad (3.55)$$

$$= -2\oint H\overline{(A_m \frac{\partial v}{\partial y})}\, dx + \oint H[\overline{A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]\, dy.$$

Let us define

$$USH = \overline{A_m \frac{\partial u}{\partial x}}, \quad UVSH = \overline{A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}, \quad \text{and } VSH = \overline{A_m \frac{\partial v}{\partial y}}, \qquad (3.56)$$

where u and v are the x and y components of the velocity output from the 3-D model. At each σ level in a triangle cell, they can be expressed as a linear function as

$$u_{i,k}(x', y') = u_{i,k}(0,0) + a^u_{(i,k)}x' + b^u_{(i,k)}y', \quad v_{i,k}(x', y') = v_{i,k}(0,0) + a^v_{(i,k)}x' + b^v_{(i,k)}y'. \qquad (3.57)$$

Then at the triangle cell i, we have

$$USH(i) = \overline{A_m \frac{\partial u}{\partial x}} = \sum_{k=1}^{KB-1} A_m(k)a^u_{(i,k)}, \qquad (3.58)$$

$$VSH(i) = \overline{A_m \frac{\partial v}{\partial y}} = \sum_{k=1}^{KB-1} A_m(k)b^v_{(i,k)}. \qquad (3.59)$$

$$UVSH(i) = \overline{A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})} = \sum_{k=1}^{KB-1} A_m(k)[a^u_{(i,k)} + b^v_{(i,k)}] \ . \qquad (3.60)$$

Therefore,

$$\overline{VISCX} = 2\oint H\overline{(A_m \frac{\partial u}{\partial x})}\, dy - \oint H[\overline{A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]\, dx$$

$$= \sum_{m=1}^{3} \overline{H}_m [USH(i) + USH(NB(m))][Y_n(N_i(j_1)) - Y_n(N_i(j_2))] \qquad (3.61)$$

$$+ 0.5\sum_{m=1}^{3} \overline{H}_m [UVSH(i) + UVSH(NB(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))];$$

$$\overline{VISCY} = -2\oint H\overline{(A_m \frac{\partial v}{\partial y})}\, dx + \oint H[\overline{A_m(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})}]\, dy$$

$$= \sum_{m=1}^{3} \overline{H}_m [VSH(i) + VSH(NB(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))] \qquad (3.62)$$

$$+ 0.5\sum_{m=1}^{3} \overline{H}_m [UVSH(i) + UVSH(NB(m))][Y_n(N_i(j_1)) - Y_n(N_i(j_2))].$$

3.2.2. The 3-D Internal Mode. The momentum equations are solved numerically using a simple combined explicit and implicit scheme in which the local change of the currents is integrated using a second-order accurate upwind scheme in time, in which the advection terms are computed explicitly by a second-order accurate upwind scheme (Kobayashi et al., 1999) and the vertical diffusion is solved implicitly. The procedure for this method is very similar to that described above for the 2-D external mode. A brief description of the numerical procedure is given as follows.

The 3-D momentum equations can be rewritten as:

$$\frac{\partial uD}{\partial t} + R_u = \frac{1}{D}\frac{\partial}{\partial \sigma}(K_m \frac{\partial u}{\partial \sigma}), \frac{\partial vD}{\partial t} + R_v = \frac{1}{D}\frac{\partial}{\partial \sigma}(K_m \frac{\partial v}{\partial \sigma}),$$ (3.63)

where

$$R_u = ADVU3 + CORX3 + DPBPX3 + BPBCX3 + HVISCX \ , $$ (3.64)

$$R_v = ADVV3 + CORY3 + DPBPY3 + BPBCY3 + HVISCY \ . $$ (3.65)

The numerical integration is conducted in two steps. In the first step, the "transition" velocity is calculated using all the terms except the vertical diffusion term in the momentum equations. Then the true velocity is determined implicitly using a balance between the local change of the "transition" velocity and the vertical diffusion term.

Let $u^*_{i,k}$ and $v^*_{i,k}$ be the x and y components of the "transition" velocity at the mid-point between the k and $k+1$ σ-levels in triangular cell i. They can be determined numerically as follows:

$$u^*_{i,k} = u^n_{i,k} - \frac{\Delta t_I}{\Omega_i \Delta \sigma \overline{D}_i} R^n_{u,(i,k)}, \quad v^*_{i,k} = v^n_{i,k} - \frac{\Delta t_I}{\Omega_i \Delta \sigma \overline{D}_i} R^n_{v,(i,k)} \ ,$$ (3.66)

where $\Delta \sigma = \sigma_k - \sigma_{k+1}$, and Δt_I is the time step for the internal mode. Each term in $R^n_{u,(i,k)}$ and $R^n_{v,(i,k)}$ is computed as follows:

$$ADVU3^n_{(i,k)} = \iint [\int_{\sigma_{k+1}}^{\sigma_k} (\frac{\partial u^2 D}{\partial x} + \frac{\partial uvD}{\partial y} + \frac{\partial u\omega}{\partial \sigma})d\sigma]dxdy$$

$$= (\sigma_k - \sigma_{k+1})\sum_{m=1}^{3} u^n_{i,k}(m)\overline{D}_m v^n_{n,k}(m)\hat{l}_m + \Omega_i[(u^n_{i,k-1} + u^n_{i,k})\omega^n_{i,k} - (u^n_{i,k} + u^n_{i,k+1})\omega^n_{i,k+1}];$$ (3.67)

$$ADVV3^n_{(i,k)} = \iint [\int_{\sigma_{k+1}}^{\sigma_k} (\frac{\partial uvD}{\partial x} + \frac{\partial v^2 D}{\partial y} + \frac{\partial v\omega}{\partial \sigma})d\sigma]dxdy$$

$$= (\sigma_k - \sigma_{k+1})\sum_{m=1}^{3} v^n_{i,k}(m)\overline{D}_m v^n_{n,k}(m)\hat{l}_m + \Omega_i[(v^n_{i,k-1} + v^n_{i,k})\omega^n_{i,k} - (v^n_{i,k} + v^n_{i,k+1})\omega^n_{i,k+1}];$$

(3.68)

$$CORX3 = -fv_i\overline{D}_i(\sigma_k - \sigma_{k+1})\Omega_i, \ CORY3 = fu_i\overline{D}_i(\sigma_k - \sigma_{k+1})\Omega_i \ ; \tag{3.69}$$

$$HVISCX^n_{(i,k)} = \iint(\int_\sigma^0 DF_x d\sigma)dxdy \approx \iint\{\int_\sigma^0 \frac{\partial}{\partial x}[2A_m H\frac{\partial u}{\partial x}] + \frac{\partial}{\partial y}[A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})]d\sigma\}dxdy$$

$$= [2\oint A_m H\frac{\partial u}{\partial x}dy - \oint \overline{A}_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})dx](\sigma_k - \sigma_{k+1})$$

$$= \{\sum_{m=1}^{3} 0.5\overline{H}_m[A_m(i) + A_m(NB(m))](a^u_{(i,k)} + a^u_{(NB(m),k)}][Y_n(N_i(j_1)) - Y_n(N_i(j_2))]$$

(3.70)

$$+ \sum_{m=1}^{3} 0.25\overline{H}_m(b^u_{(i,k)} + b^u_{(NB(m),k)} + a^v_{(i,k)} + a^v_{(NB(m),k)}]$$

$$[X_n(N_i(j_2)) - X_n(N_i(j_1))][A_m(i) + A_m(NB(m))]\}(\sigma_k - \sigma_{k+1});$$

$$HVISCY^n_{(i,k)} = \iint(\int_\sigma^0 DF_y d\sigma)dxdy \approx \iint\{\int_\sigma^0 \frac{\partial}{\partial y}[2A_m H\frac{\partial v}{\partial y}] + \frac{\partial}{\partial x}[A_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})]d\sigma\}dxdy$$

$$= [-2\oint A_m H\frac{\partial v}{\partial y}dx + \oint \overline{A}_m H(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})dy](\sigma_k - \sigma_{k+1})$$

$$= \{\sum_{m=1}^{3} 0.5\overline{H}_m[A_m(i) + A_m(NB(m))](b^v_{(i,k)} + b^v_{(NB(m),k)}][X_n(N_i(j_2)) - X_n(N_i(j_1))]$$

(3.71)

$$+ \sum_{m=1}^{3} 0.25\overline{H}_m(b^u_{(i,k)} + b^u_{(NB(m),k)} + a^v_{(i,k)} + a^v_{(NB(m),k)}]$$

$$[Y_n(N_i(j_1)) - Y_n(N_i(j_2))][A_m(i) + A_m(NB(m))]\}(\sigma_k - \sigma_{k+1});$$

$$DPBPX^n_{(i,k)} = g\overline{D}_i(\sigma_k - \sigma_{k+1})\sum_{m=1}^{3} \overline{\zeta}^n_m[Y_n(N_i(j_1)) - Y_n(N_i(j_2))]; \tag{3.72}$$

$$DPBPY^n_{(i,k)} = g\overline{D}_i(\sigma_k - \sigma_{k+1})\sum_{m=1}^{3} \overline{\zeta}^n_m[X_n(N_i(j_2)) - X_n(N_i(j_1))]; \tag{3.73}$$

$$PBCX3 = -\frac{g}{\rho_o}\{\iint \int_{\sigma_{k+1}}^{\sigma_k} D\frac{\partial}{\partial x}[D(\int_\sigma^0 \sigma\frac{\partial \rho}{\partial \sigma}d\sigma)]d\sigma dxdy + \iint \int_{\sigma_{k+1}}^{\sigma_k} D^2\frac{\partial \rho\sigma}{\partial x}d\sigma dxdy\}$$

$$= -\frac{g}{\rho_o}\{\overline{D}_i\oint(D\int_\sigma^0 \sigma\frac{\partial \rho}{\partial \sigma}d\sigma)dy + \overline{D_i^2}\oint \rho\sigma dy\}[\sigma_k - \sigma_{k+1}]$$

(3.74)

Let

31

$$PBC(i) = \int_{\sigma}^{0} \sigma \frac{\partial \rho}{\partial \sigma} d\sigma = \sum_{k'=1}^{k} \sigma(k')[\rho(k') - \rho(k'+1)], \tag{3.75}$$

then

$$DPBCX = -\frac{0.5g}{\rho_o}(\sigma_k - \sigma_{k+1})$$

$$\{\overline{D}_i \sum_{m=1}^{3} \overline{D}_m [PBC(i) + PBC(NB_i(m))][Y_n(N_i(j_1)) - Y_n(N_i(j_2))] \tag{3.76}$$

$$+\overline{D_i^2} \sum_{m=1}^{3} [\rho(i) + \rho(NB_i(m))]\sigma(k)[Y_n(N_i(j_1)) - Y_n(N_i(j_2))]\};$$

Similarly, we can derive the y-component of the baroclinic pressure gradient as

$$DPBCY = -\frac{0.5g}{\rho_o}(\sigma_k - \sigma_{k+1})$$

$$\{\overline{D}_i \sum_{m=1}^{3} \overline{D}_m [PBC(i) + PBC(NB_i(m))][X_n(N_i(j_2)) - X_n(N_i(j_1))] \tag{3.77}$$

$$+\overline{D_i^2} \sum_{m=1}^{3} [\rho(i) + \rho(NB_i(m))]\sigma(k)[X_n(N_i(j_2)) - X_n(N_i(j_1))]\}.$$

The mathematic forms of the two equations in (3.63) are the same, so that they can be solved numerically using the same approach. The method used to numerically solve these equations was adopted directly from ECOM-si and POM (Blumberg, 1994). For example, a detailed description of this method is given below for the *u*-component of the momentum equation. The implicit discrete form of the first equation in (3.63) is given as

$$A_{i,k} u_{k+1}^{n+1} + B_{i,k} u_{k}^{n+1} + C_{i,k} u_{k-1}^{n+1} = u^{*} \tag{3.78}$$

where

$$A_{i,k} = -\frac{2K_m(k+1)\Delta t}{[D^{n+1}]^2(\sigma_k - \sigma_{k+1})(\sigma_k - \sigma_{k+2})};$$

$$C_{i,k} = -\frac{2K_m(k)\Delta t}{[D^{n+1}]^2(\sigma_k - \sigma_{k+1})(\sigma_{k-1} - \sigma_{k+1})}; \tag{3.79}$$

$$B_{i,k} = 1 - A_{i,k} - C_{i,k}.$$

This is a tri-diagonal equation and it ranges from $k = 2$ to KB-2, where KB is the number of total σ-levels in the vertical. The solution for $u(k)$ is calculated by

$$u(k) = -\frac{A_{i,k}}{B_{i,k} + C_{i,k}VH(k-1)}u(k+1) + \frac{u^* - C_{i,k}VHP(k-1)}{B_{i,k} + C_{i,k}VH(k-1)} \quad (3.80)$$

where

$$VH(k) = -\frac{A_{i,k}}{B_{i,k} + C_{i,k}VH(k-1)}; \; VHP(k) = \frac{u^*_{i,k} - C_{i,k}VHP(k-1)}{B_{i,k} + C_{i,k}VH(k-1)} \; . \quad (3.81)$$

The equation for temperature or salinity as well as other passive tracers also can be rewritten as the form shown in (3.78), so they can be solved numerically using the exact same approach discussed above. The only difference is that T is calculated at nodes and has the same control volume as that used for ζ. To apply a second-order upwind scheme for the temperature advection term, we used Green's theorem to calculate the temperature
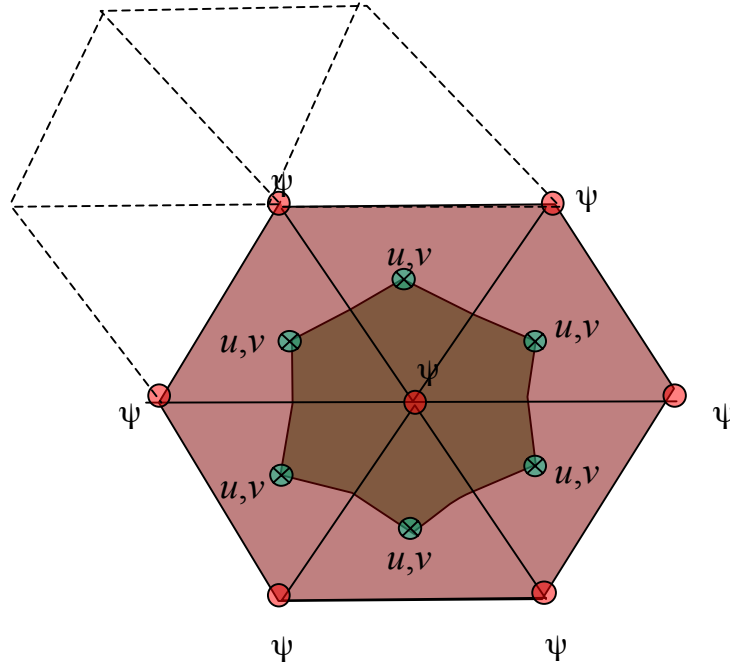
Fig. 3.4: Illustration of the second-order upwind scheme used to calculate the horizontal advection terms in a tracer equation.

gradient at nodes (Barth, 1993; Wu and Bogy, 2000). Computing T at nodes has shown a significant improvement in the advective temperature flux over steep bottom topography.

In the original version of FVCOM, the horizontal advection terms in a tracer equation are calculated using the second-order accurate upwind scheme shown in Fig. 3.4. Let ψ

be an arbitrary tracer variable (which can be *T*, *S* or *q* or others), the value of ψ at the centroid of a triangle can be determined by

$$\psi = \psi_i + \frac{\partial \psi}{\partial \lambda} \Delta \lambda + \frac{\partial \psi}{\partial \phi} \Delta \psi$$

$$\frac{\partial \psi}{\partial \lambda} = \frac{r}{\Omega_{\psi_i}} \oint \psi d\phi; \frac{\partial \psi}{\partial \phi} = \frac{r}{\Omega_{\psi_i}} \oint [\psi(\cos\phi)_{\psi_i}] d\lambda$$

(3.82)

where subscript "*i*" represents the index of the node over TCE and Ω_{ψ_i} is the area of the larger TCE with a center node at ψ_i. In the upwind scheme, ψ at the centroid is always calculated using the flux determined with velocity and distribution of ψ in the upwind side of the control volume.

   In recent applications to submarine banks and islands where the bottom topography is steep, we found that the central difference scheme used in the vertical advection term can cause odd-even decoupling due to the unstable nature of the scheme. To retaining the second-order accuracy upwind scheme, we introduced a second-order accurate Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) with Flux Corrected Transport (FCT) as an option for the calculation of the vertical advection term (Smolarkiewicz, 1984; Smolarkiewicz and Grabowski, 1990; Smolarkiewicz and Szmelter, 2005). This work was done in 2007_by Song Hu, then a graduate student in the MEDM Laboratory at SMAST, who applied FVCOM to study the tidal pumping process on the steep flanks of Georges Bank (Hu et al., 2008). This approach provides an optimal control of an artificial numerical error due to over-or-under-shooting of the tracer value over steep bottom topography. FCT ensures a second-order accurate positive definite tracer calculation in a 3-D volume away from discontinuties. A brief description of the algorithm is given here.

   The 3-D advection equations can be written as

$$\frac{\partial \psi}{\partial t} + HADV + VADV = 0$$

(3/83)

where *HADV* and *VADV* are the horizontal and vertical advection terms in the tracer

equation, respectively. Define $\psi_{i,k}^{n+1}$ as the value of ψ at the ith node and kth vertical

level at the time step $n+1$. Then $\psi_{i,k}^{n+1}$ can be calculated following two steps.

Step 1:

$$\psi_{i,k}^{(*)^0} = \psi_{i,k}^n - \min(1, \beta_{i,k})\Psi_{i,k}^n, \tag{3.84}$$

where

$$\Psi_{i,k}^n = (1 - \alpha_i)\psi_{i,k}^n + \alpha_i \Delta t (HADV + VDAV) \ , \tag{3.85}$$

$$\beta_{i,k} = \begin{cases} \left[\dfrac{\psi_{i,k}^n - \psi_{i,k}^{\min}}{\Psi_{i,k}^n}\right]^+ - \left[\dfrac{\psi_{i,k}^{\max} - \psi_{i,k}^n}{\Psi_{i,k}^n}\right]^- & \text{as } \Psi_{i,k}^n \neq 0 \\[4mm] 0 & \text{as } \Psi_{i,k}^n = 0 \end{cases}, \tag{3.86}$$

$$F_v = -\frac{1}{2\Delta\sigma}\{[\omega_{i,k}^n]^- \psi_{i,k-1}^n + [\omega_{i,k}^n]^+ \psi_{i,k}^n - [\omega_{i,k+1}^n]^- \psi_{i,k}^n - [\omega_{i,k+1}^n]^+ \psi_{i,k+1}^n\}. \tag{3.87}$$

Step 2:

$$\begin{aligned} \psi_{i,k}^{(*)m} = \psi_{i,k}^{(*)^{m-1}} \\ -\frac{\Delta t}{2\Delta\sigma}\Big\{[\omega_{i,k}^{(*)^m}]^{mon^-}\psi_{i,k-1}^{(*)^{m-1}} + [\omega_{i,k}^{(*)^m}]^{mon^+}\psi_{i,k}^{(*)^{m-1}} - [\omega_{i,k+1}^{(*)^m}]^{mon^-}\psi_{i,k}^{(*)^{m-1}} - [\omega_{i,k+1}^{(*)^m}]^{mon^+}\psi_{i,k+1}^{(*)^{m-1}}\Big\} \end{aligned} \tag{3.88}$$

$$\psi_{i,k}^{n+1} = \psi_{i,k}^{(*)^{IORD}}, \tag{3.89}$$

where m is the iteration number from 1 to IORD. The default value of IORD is 4. For any

arbitrary function Φ, $[\Phi]^- = \min(0, \Phi)$ and $[\Phi]^+ = \max(0, \Phi)$. The terms related to ω in

(3.88) are defined as

$$[\omega_{i,k}^{(*)^m}]^{mon} = \min(1, \beta_{i,k-1}^\uparrow, \beta_{i,k}^\downarrow)[\omega_i^{(*)^m}]^+ + \min(1, \beta_{i-1}^\downarrow, \beta_i^\uparrow)[\omega_i^{(*)^m}]^-, \tag{3.90}$$

$$\beta_{i,k}^\downarrow = \frac{\psi_{i,k}^{(*)^{m-1}} - \psi^{\min}}{\dfrac{\Delta t}{\Delta\sigma}\{[\omega_{i,k}^{(*)^m}]^+ \psi_{i,k}^{(*)^{m-1}} - [\omega_{i,k+1}^{(*)^m}]^- \psi_{i,k}^{(*)^{m-1}}\} + \varepsilon}, \tag{3.91}$$

$$\beta_{i,k}^{\uparrow} = \frac{\psi^{max} - \psi_i^{(*)^{k-1}}}{\frac{\Delta t}{\Delta \sigma}\{[\omega_{i+1}^{(*)^m}]^+ \psi_{i,k+1}^{(*)^{m-1}} - [\omega_{i,k}^{(*)^m}]^- \psi_{i,k-1}^{(*)^{m-1}}\} + \varepsilon}, \tag{3.92}$$

$$w_i^{(*)^m} = \frac{\{\Delta \sigma \mid w_{i,k}^{(*)^{m-1}} \mid - \Delta t \mid \omega_{i,k}^{(*)^{m-1}} \mid^2\}(\psi_{i,k-1}^{(*)^{m-1}} - \psi_{i,k}^{(*)^{m-1}})}{(\psi_{i,k-1}^{(*)^{m-1}} - \psi_{i,k}^{(*)^{m-1}} + \varepsilon)\Delta \sigma} \tag{3.93}$$

where ε is a small value set up as 10^{-10}, $\omega_{i,k}^{(*)0} = \omega_{i,k}^n$.

3.3. Transport Consistency of External and Internal Modes

In a mode-split model, an adjustment must be made in every internal time step to ensure the consistency in the vertically integrated water transport produced by external and internal modes (Chen et al., 2004b). In FVCOM, the vertical velocity at the σ surface (ω) is calculated by

$$\omega_{i,k+1} = \omega_{i,k} + \frac{\Delta \sigma_k}{\Delta t_I}(\zeta_i^{n+1} - \zeta_i^n) + \frac{\Delta \sigma_k}{\Omega_i}\oint_l v_{N,k}^n D dl, \tag{3.94}$$

where i, k, and n is indicators of the ith node point, kth σ level and nth time step, respectively; Ω_i^n is the area of the ith TCE at the nth time step. N is an indicator of the velocity component normal to the boundary of a TCE with a length of l. To conserve the volume on the ith TCE, the vertically integrated form of eq. (3.94) must satisfy

$$\frac{\zeta_i^{n+1} - \zeta_i^n}{\Delta t_I} + \sum_{k=1}^{kb-1} \frac{\Delta \sigma_k}{\Omega_i^n}\oint_l v_{N,k}^n D dl = 0, \tag{3.95}$$

where kb is the total number of the σ levels. Since ζ_n^{n+1} is calculated through the vertically integrated continuity equation over I_{split} external mode time steps (where $I_{split} = \dfrac{\Delta t_I}{\Delta t_E}$), Eq. (3.95) is valid only if

$$\Delta t_I \sum_{k=1}^{kb-1} \frac{\Delta \sigma_k}{\Omega_i^n}\oint_l v_{N,k}^n D dl = \Delta t_E \sum_{\hat{n}=1}^{I_{split}} \frac{1}{\Omega_i^{\hat{n}}}\oint_l \bar{v}_N^{\hat{n}} D dl. \tag{3.96}$$

Because the numerical accuracy depends on the time step, the left and right sides of Eq. (3.96) is not exactly equal to each other unless $I_{split} = 1$. Therefore, to ensure the conservation of the volume transport throughout the water column of the ith TCE, the

internal velocity in each σ-layer must be calibrated using the difference of an inequality of vertically integrated external and internal water transport before ω is calculated. Since the vertically integrated transport features the barotropic motion, the easiest calibration way is to adjust the internal velocity by distributing the "error" uniformly throughout the water column. ω, which is calculated with adjusted internal velocities through Eq. (3.94), not only guarantees that the volume transport is conservative in the whole water column but also in an individual TCE volume in each σ layer. Because the water temperature and salinity or other tracers are calculated in the same TCE volume as that used for ω, this transport adjustment also conserves the mass in an individual TEC volume in each σ layer.

3.4. The Wet/Dry Treatment Technique

One of the most difficult problems in estuarine modeling is to provide an accurate simulation of the water transport flooding onto and draining out of the inter-tidal zone, which can be several times as large as the transport in the main channel of a river. In the last three decades, two types of approaches have been developed to solve this issue: one is the moving-boundary method and the other is the wet/dry point treatment method. In the first method, the computational domain is bounded by an interface line between land and water where total water depth and normal transport are equal to zero (Lynch and Gray, 1980). Because this boundary moves over flooding and draining cycles, the model grid must be re-generated at every time step. This method seems to work for idealized estuaries with simple geometries (Sidén and Lynch, 1988; Austria and Aldama, 1990), but it is not practical for application to realistic estuaries with complex geometry including tidal creeks, islands, barriers, and inlets. In the second method, the computational domain covers the maximum flooding area. Numerical grids consist of wet and dry points with a boundary defined as an interface line between water and land respectively. The wet and dry points are distinguished by the local total water depth of $D = H(x,y) + \zeta(x,y,t)$ (where H is the reference water depth and ζ is the surface elevation). The wet point is a grid point with $D > 0$, otherwise, $D = 0$ at dry points. At dry points, velocities are automatically specified as zero, but salinity retains the same

value from the previous time step. Since this method is relatively simple, it has been widely used to simulate the water transport over inter-tidal areas in estuarine models (Leendertse, 1970 & 1987; Flather and Heaps, 1975; Ip et al., 1998, Zheng et al., 2003a).

The wet/dry point treatment technique works only for the case in which a finite-value solution of the governing equations exists as D approaches zero. In a z-coordinate system, to ensure numerical stability of a 3-D model, the thickness of the layer closest to the surface must be greater than the amplitude of the tidal elevation (Davis et al., 1997). This makes it difficult to apply this method for a shallow estuary in which the amplitude of the tidal elevation is the same order of magnitude as the local water depth. Because irregular bathymetry is poorly resolved in a z-coordinate model, it is not likely that the water transport can be accurately simulated over realistic bathymetric inter-tidal zones during both flooding and draining periods. Examples of the z-coordinate wet/dry methods can be seen in Casulli and Cheng (1991 & 1992), Cheng et al. (1993), Casulli and Cattani (1994), and Hervouet and Janin (1994).

In a σ-coordinate system, the wet/dry point treatment is no longer valid as D becomes zero. One simple way to avoid numerical singularity is to specify zero velocities at all dry points. This approach, however, cannot ensure volume conservation in the numerical computation due to discontinuous water removal from elements that turn to dry over one time step. An alternative way is to add a viscous boundary layer (h_c) at the bottom and redefine wet/dry points using a sum of D and h_c. The grid is treated as a wet point for $D > h_c$, otherwise it is a dry point. In terms of the nature of the vertical structure of turbulent mixing, a viscous layer always exists below the log boundary layer near a solid wall (Wilcox, 2000). However, to avoid adding additional water transport into a dynamic system, the viscous layer should be sufficiently small to satisfy a motionless condition. Good examples of the application of this method can be found in Ip et al. (1998) and Zheng et al. (2003b).

No matter which methods are used to simulate the flooding/draining process over the intertidal zone in an estuary, they must be validated with respect to mass conservation. In all of these methods, the dry and wet points are determined using some empirical criteria, so that the estimation of the water transport in the dry-wet transition zone depends on 1)

the criterion used to define the wet/dry points, 2) the time step used for numerical integration, 3) the horizontal and vertical resolutions of the model grids, 4) amplitudes of surface elevation, and 5) bathymetry. In a σ-coordinate transformation model, it might be also related to the thickness of the bottom viscous layer (D_{min}).

A new wet/dry point treatment method has been developed for use with FVCOM (see Chen et al., 2006c-unpublished manuscript). This method has been validated in a series of tidal simulations using an idealized semi-enclosed estuary with an inter-tidal zone. Relationships of the time step with discrete grid resolution, amplitude of external forcing, the slope of the inter-tidal zone and thickness of the bottom viscous layer are discussed and the criterion for the selection of the time step has been derived. The rule used in validation is mass conservation, which, we believe, is a prerequisite condition for an objective evaluation of the wet/dry point treatment technique in estuaries (and coastal regions where inundation occurs).

3.4.1　Criteria. Defining that $D = H_m + \zeta$, where $H_m = H + D_{min}$ in the water and $H_m = -(h_B + D_{min})$ on the land**,** the wet or dry criterion for node points is given as

$$\begin{cases} \text{wet,} & \text{if } D = H_m + \zeta > D_{min} \\ \text{dry} & \text{if } D = H_m + \zeta \le D_{min} \end{cases} \tag{3.97}$$

and for triangular cells is given as

$$\begin{cases} \text{wet,} & \text{if } D = \min(H_{m,i}, H_{m,\hat{j}}, H_{m,\hat{k}}) + \max(\zeta_{\hat{i}}, \zeta_{\hat{j}}, \zeta_{\hat{k}}) > D_{min} \\ \text{dry,} & \text{if } D = \min(H_{m,i}, H_{m,\hat{j}}, H_{m,\hat{k}}) + \max(\zeta_{\hat{i}}, \zeta_{\hat{j}}, \zeta_{\hat{k}}) \le D_{min} \end{cases} \tag{3.98}$$
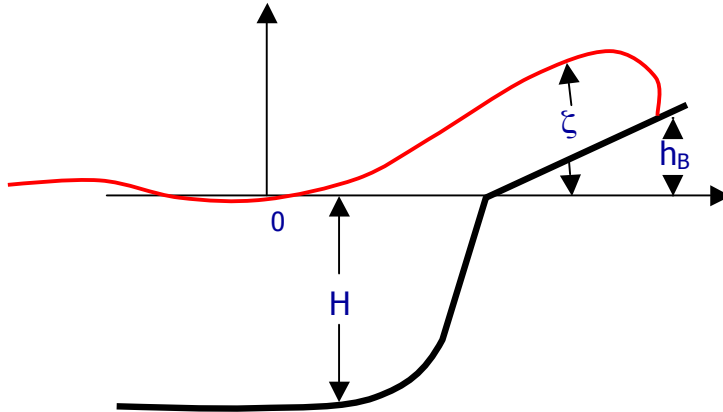
Fig. 3.5: Definition of reference depth (H), surface level (ζ) and bathymetric height (h_B).

where D_{min} is the thickness of the viscous layer specified at the bottom, h_B is the bathymetric height related to the edge of the main channel of a river (Fig.3.5) and $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the integer numbers to identify the three node points of a triangular cell.

When a triangular cell is treated as dry, the velocity at the centroid of this triangle is specified to be zero and no flux is allowed on the three boundaries of this triangle. This triangular cell is removed from the flux calculation in the TCE. For example, the integral form of the continuity equation in FVCOM is written as

$$\iint_{TCE} \frac{\partial \zeta}{\partial t} \, dxdy = -\iint_{TCE} [\frac{\partial (\bar{u}D)}{\partial x} + \frac{\partial \bar{v}D}{\partial y}] dxdy = -\oint_{l} \bar{v}_N Ddl \qquad (3.99)$$

where $\bar{u}$ and $\bar{v}$ are the x and y components of the vertically-averaged velocity. In a dry/wet point system, only wet triangles are taken into account in the flux calculation in a TCE since the flux on boundaries of the dry triangle is zero (see Fig. 3.6). This approach always ensures the volume conservation in a TCE that contains the moving boundary between the dry and wet triangles over an integration interval. The same approach is used to calculate the tracer flux (temperature, salary and other scalar tracers) and momentum flux in a MCE.
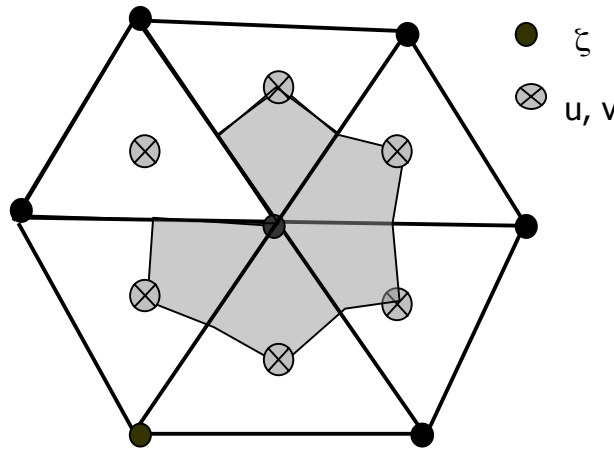
Fig. 3.6: Illustration of dye/wet triangles within a TCE.

One of the critical issues in applying the wet/dry point treatment technique in a split-mode model is to ensure mass conservation in the individual TCE which is crossed by the

moving boundary. Because $\Omega_i^{\hat{n}}$ may change within I_{split} external time integrations due to the occurrence of dry triangles and ζ is treated as zero when D is less than D_{min}, we have:

$$\zeta_i^{n+1} - \zeta_i^n \neq \Delta t_E \sum_{n=1}^{I_{split}} \frac{1}{\Omega_i^{\hat{n}}} \oint_l \bar{v}_N^{\hat{n}} D dl . \tag{3.100}$$

In this case, the external and internal mode adjustment through Eq. (3.96) can not guarantee that ω reaches zero at $\sigma = -1$ for the internal mode. To ensure the volume conservation, an additional adjustment for ζ_i^{n+1} must be made in the TCE when ω is calculated by Eq.(3.95).

The additional sea level adjustment works in general, but fails in the case where ζ is very close to D_{min} (for example, $\Delta\zeta = \zeta - D_{min} < 10^{-1}$ m). When this happens, small errors in the calculation of volume flux can rapidly accumulate through nonlinear feedbacks of tracer advection and eventually destroy the nature of the mass conservation.

FVCOM always ensures the mass conservation in the wet-dry transition zone if $I_{split} = 1$. For computational efficiency, however, we want to find an approach so that mass conservation is still guaranteed for the case in which $I_{split} > 1$. In additional to the criterion of general numerical instability, in a case with inclusion of the

Fig. 3.7: The model-predicted relationship of the ratio of the internal to external mode time steps ($\Delta t_I / \Delta t_E$) with the tidal forcing amplitude (ζ_o) and the bottom slope of the inter-tidal zone (α). In these experiments, $\Delta t_E = 4.14$ sec, $kb = 6$, $D_{min} = 5$ cm.

flooding/drying process, the choice of I_{split} is restricted by many other factors including the surface elevation, bathymetry, and thickness of the bottom viscous layer and

horizontal/vertical resolutions. A discussion on the relationship of I_{split} to these factors is given next through numerical experiments for idealized cases.

3.4.2. The upper-bound limit of I_{split}. By simulating the flooding/drying process in an idealized semi-enclosed channel with an inter-tidal zone, we examined the relationship of the model forcing and geometric parameters on the upper-bound limit of I_{split} (Chen et al., 2006b). A brief summary of the model results is given below.
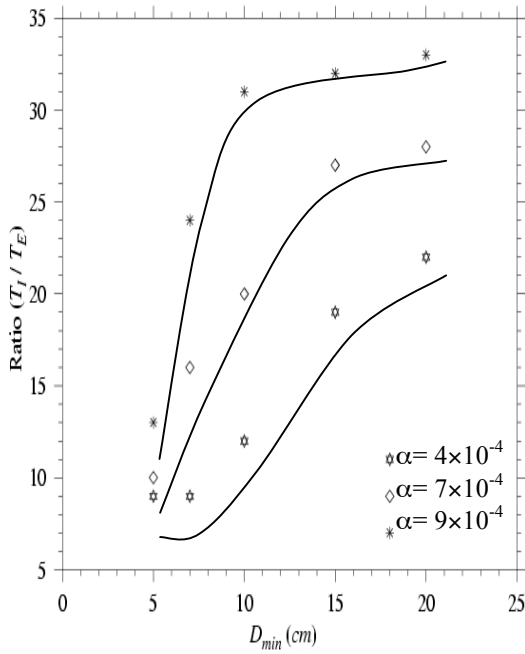
Figure 3.8: The model-derived relationship of I_{split} with D_{min} for the three cases with $\alpha = 4.0\times10^{-4}$, 7.0×10^{-4} and 9.0×10^{-4}. In the three cases, $\Delta t_E = 4.14$ sec, kb = 6, and $D_{min} = 1.5$ m.
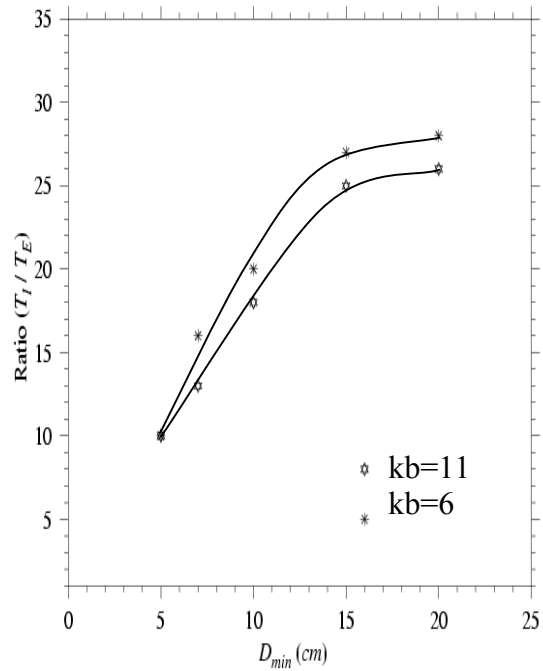
Figure 3.9: The model-derived relationship of I_{split} with D_{min} for the two cases with kb = 6 and 11, respectively. In these two cases, $\Delta t_E = 4.14$ sec, $\Delta t_E = 4.0\times10^{-4}$, and $\zeta_o = 1.5$ m.

The relationship with α and ζ_o. The model results show that the upper-bound value of I_{split} varies with the bottom slope of the inter-tidal zone (α) and amplitude of tidal forcing (ζ_o) (Fig. 3.7). Considering a standard case with a constant slope of $\alpha = 4.0\times10^{-4}$, the upper-bound value of I_{split} gradually becomes smaller as ζ_o becomes larger. It is below 10 at ζ_o=2.0 m and up to 15 at ζ_o=0.5 m. When the slope is up to 7.0×10^{-4}(a change in the height of the inter-tidal zone up to 1.4 m over a distance of 2

km), however, the upper-bound value of I_{split} dramatically increases in a tidal forcing range of $\zeta_o < 1.0$ m, even though it remains only slightly higher than the standard case with larger tidal forcing. The model still conserves mass in the wet-dry transition zone at $I_{split} = 70$ at $\zeta_o = 0.5$ m. I_{split} becomes even more flexible in the case with steeper slope of $\alpha = 9.0 \times 10^{-4}$ (a change in the height of the inter-tidal zone up to 1.8 m over a distance of 2 km). In the tidal forcing range of $\zeta_o < 1.5$ m, the upper bound value of I_{split} increases almost exponentially with the decrease of ζ_o. Even in the larger tidal forcing range of $\zeta_o > 1.5$ m, the upper-bound value of I_{split} exceeds 10.

The relationship with D_{min}. In general, under given tidal forcing, vertical/horizontal resolutions, and external mode time step, the upper-bound value of I_{split} increases as D_{min} becomes larger (Fig. 3.8). In the standard case with $\zeta_o = 1.5$ m and $\alpha = 4.0 \times 10^{-4}$, for example, I_{split} must be smaller or equal to 9 for the case with $D_{min} = 5$ cm, but it could be 22 for the case with $D_{min} = 20$ cm. I_{split} could be much larger in the case with a steeper slope of the inter-tidal zone. In the cases with $\alpha = 7.0 \times 10^{-4}$ and 9.0×10^{-4}, the upper-bound value of I_{split} could
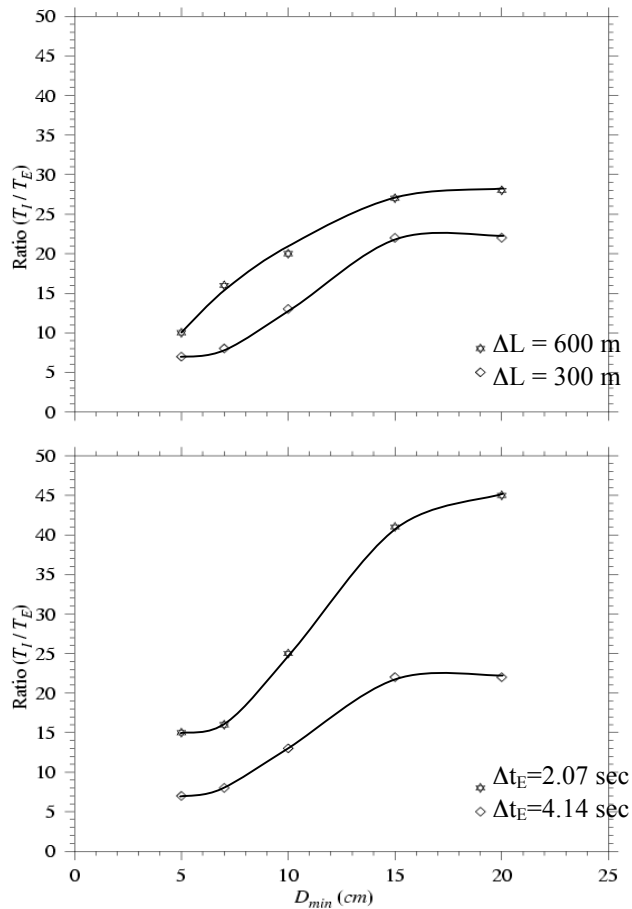
Fig. 3.10:　The model-derived relationship of I_{split} with ΔL (upper panel) and Δt_E (lower panel). In the upper panel case, $\Delta t_E = 4.14$ sec, $\alpha = 7.0 \times 10^{-4}$, kb =6, and $\zeta_o = 1.5$ m. In the lower panel case, $\Delta L = 300$ m, $\alpha = 7.0 \times 10^{-4}$, kb =6, and $\zeta_o = 1.5$ m.

be up to 10 and 13, respectively for $D_{min} = 5$ cm and up to 28 and 33, respectively for $D_{min} = 20$ cm.

The relationship with kb (# of sigma levels). The upper-bound limit of I_{split} with respect to vertical resolution is sensitive to the thickness (D_{min}) of the viscous layer specified in the model (Fig. 3.9). For a standard case with kb = 6 and $\alpha = 7.0 \times 10^{-4}$, for example, with a tidal forcing of $\zeta_o = 1.5$ m, the upper-bound value of I_{split} is 10 at $D_{min} = 5$ cm and up to 28 at $D_{min} = 20$ cm. Keeping the same forcing condition but increasing kb to 11, we found that the upper-bound value of I_{split} remains almost the same with $D_{min} = 5$ cm but drops significantly as D_{min} increases.

The relationship with ΔL **and** Δt_E. For a given Δt_E, the upper-bound value of I_{split} decreases as horizontal resolution increases (Fig. 3.10). For a standard case, for example, with a tidal forcing of $\zeta_o = 1.5$ m, the upper-bound limit of I_{split} is cutoff at 10 for $D_{min} = 5$ cm and at 28 for $D_{min} = 20$ cm. These values, however, drop to 7 for $D_{min} = 5$ cm and to 21 for $D_{min} = 20$ cm when ΔL decreases to 300 m (Fig. 3.10: upper panel). Similarly, for a given $\Delta L = 300$ m, the upper-bound value of I_{split} increases significantly as Δt_E decreases, which jumps from 7 to 15 at $D_{min} = 5$ cm and from 21 to 45 at $D_{min} = 20$ cm as Δt_E decreases from 4.14 sec to 2.07 sec (Fig.3.10: lower panel). In the range of D_{min} shown in Fig. 3.10, for the two cases with $(\Delta t_E)_1$ and $(\Delta t_E)_2$, $(I_{split})_2$ is approximately estimated by

$$(I_{split})_2 \sim (I_{split})_1 \frac{(\Delta t_E)_1}{(\Delta t_E)_2} .$$

It should be noted here that the actually upper-bound limit of I_{split} in application to realistic estuaries and coastal flooding areas might be different from the results presented here for our idealized test case. The key point of presenting these idealized model results is to inform users about the dependence of the upper-bound of I_{split} on tidal ranges, bottom slope in the inter-tidal zone, thickness of the specified viscous layer, external time

step, and vertical/horizontal resolution, and provide a guide for choosing I_{split} in realistic applications.

3.5. Finite-Volume Discrete Methods in Spherical Coordinate System

The numerical methods used to solve the spherical coordinate version of FVCOM are the same as those used in the Cartesian coordinate version of FVCOM with two exceptions, the redefinition of the meridian flux and North Pole treatment. In both Cartesian and spherical coordinates, we have introduced a new flux corrected second-order scheme to calculate the tracer advection. The discrete procedure of FVCOM was given in detail in Chen et al. (2003a) and Chen et al. (2004b), and brief descriptions of the re-definition of meridian flux, the discrete scheme for the tracer advection, and North Pole treatment are given below. The text is directly adopted from Chen et al. (2006a).

Following the same approach used in the Cartesian coordinate version of FVCOM,
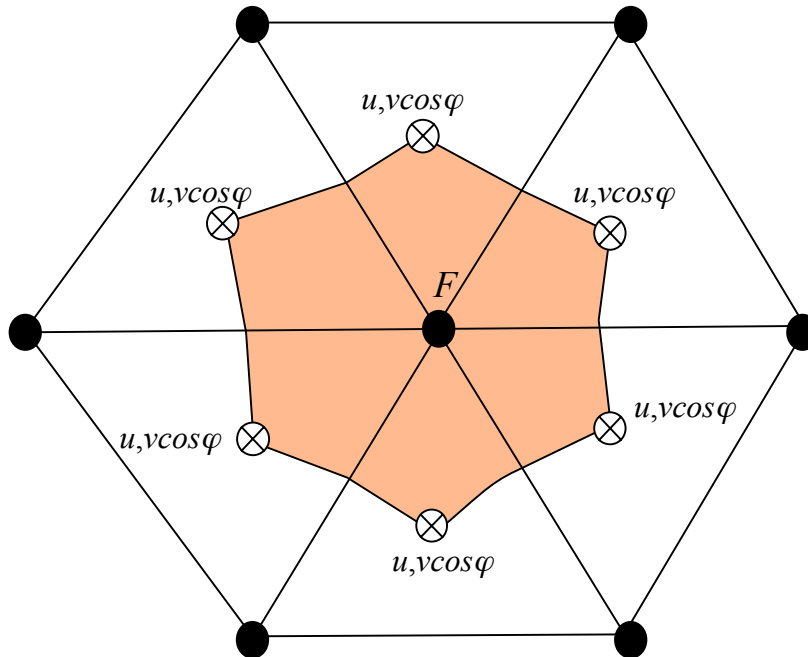
Fig. 3.11: Schematic of the control volume used to calculate scalar variables and vertical velocity used in FVCOM. *F* is a general symbol representing scalar variables such as ζ, *T*, *S*, K_m, K_h, and vertical velocity ω. • is the node of the triangles where scalar variable or vertical velocity is calculated and $\otimes$ is the centroid of a triangle where the horizontal velocity is calculated.

the horizontal numerical computational domain is subdivided into a set of non-

overlapping unstructured triangular cells. An unstructured triangle is comprised of three nodes, a centroid, and three sides (Fig. 3.11), on which u and v are placed at centroids and all scalar variables, such as ζ, H, D, ω, S, T, ρ, K_m, K_h, A_m, and A_h are placed at nodes. u and v at centroids are calculated based on the net flux through three sides of that triangle (called the momentum control element: MCE), while scalar variables at each node are determined by the net flux through the sections linked to centroids and the middle point of the sideline in the surrounding triangles (called the tracer control element: TCE).

In both 2-D (external mode) and 3-D (internal mode) momentum equations, the advection term is calculated in the flux form using a second-order accurate upwind finite-difference scheme (Kobayashi et al., 1999; Hubbard, 1999, Chen et al. 2003a), which is the same as that used in the Cartesian FVCOM. In this scheme, the velocity in the
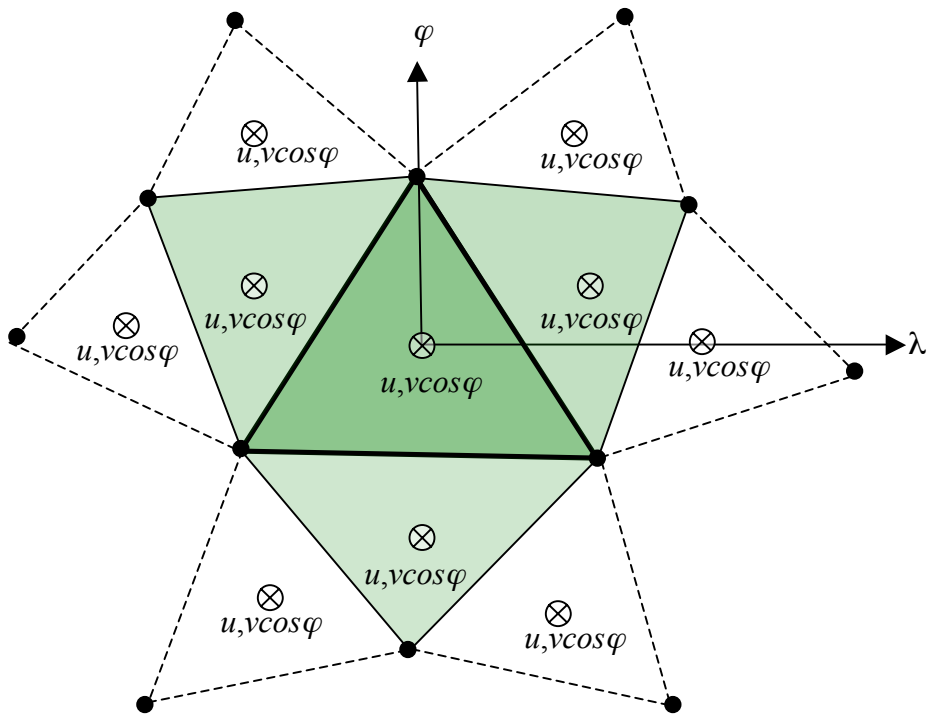
Fig. 3.12: Schematic of the momentum control volume (bounded by heavy solid lines) used to calculate the horizontal velocity. Light gray filled triangles are surrounding meshes required to solve the linear equation to determine the velocity distribution.

triangle cell i is assumed to satisfy the linear distribution given as

$$u_i(x', y') = \phi_i^u(x', y') = u_i(x_c, y_c) + a_i^u(x' - x_c) + b_i^u(y' - y_c) \tag{3.101}$$

$$v_i(x', y') = \phi_i^v(x', y') = v_i(x_c, y_c) + a_i^v(x' - x_c) + b_i^v(y' - y_c) \tag{3.102}$$

where (x_c, y_c) is the location of the center of the triangular cell i and (x', y') is the location of any point in three adjacent triangular cells. The parameters a_i^u, b_i^u, a_i^v, and b_i^v are determined by a least-square method based on velocity values at the four cell-centered points shown in Fig. 3.12. On the sphere, the curved nature of the surface needs to be taken into account when the area, lengths, and center of a triangle are calculated. The area of a triangle on a sphere equals

$$ART = r^2\delta \tag{3.103}$$

and

$$\delta = 2\arcsin\left[\frac{\sqrt{\sin p \sin(p-a)\sin(p-b)\sin(p-c)}}{2\cos\frac{a}{2}\cos\frac{b}{2}\cos\frac{c}{2}}\right] \tag{3.104}$$

where $p = \frac{1}{2}(a + b + c)$, and a, b, and c are the arc length of the three side boundaries of a triangle. The arc length (defined as $\overline{AB}$) between two points (λ_a, φ_a) and (λ_b, φ_b) is calculated as follows. The x, y and z at these two points can be given as

$$x_a = r\cos\varphi_a\cos\lambda_a, \; y_a = r\cos\varphi_a\sin\lambda_a, \; z_a = r\sin\varphi_a , \tag{3.105}$$

$$x_b = r\cos\varphi_b\cos\lambda_b, \; y_b = r\cos\varphi_b\sin\lambda_b, \; z_b = r\sin\varphi_b . \tag{3.106}$$

The "string" distance (defined as $\overline{ab}$) between these two points is equal to

$$\overline{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} . \tag{3.107}$$

According to the *cosine* theorem, the central angle α of the arc can be expressed as

$$\alpha = \arccos\frac{2r^2 - ab^2}{2r^2}. \tag{3.108}$$

Therefore, the arc length $\overline{AB}$ between point A and point B is equal to

$$\overline{AB} = r\alpha . \tag{3.109}$$

The x_c, y_c, and z_c at the string center are equal to

$$x_c = r \frac{\cos\varphi_a \cos\lambda_a + \cos\varphi_b \cos\lambda_b}{2}, \qquad (3.110)$$

$$y_c = r \frac{\cos\varphi_a \sin\lambda_a + \cos\varphi_b \sin\lambda_b}{2}, \qquad (3.111)$$

$$z_c = r \frac{\sin\varphi_a + \sin\varphi_b}{2}. \qquad (3.112)$$

The arc lengths in the x and y directions between two points (λ_a, φ_a) and (λ_b, φ_b) are given as

$$dx = r\cos\varphi d\lambda \text{ and } dy = rd\varphi. \qquad (3.113)$$

After the area of the MCE and arc lengths relative to the center of the MCE are determined, the distribution of u and v in four triangles shown in Fig. 3.12 can be determined, and the second-order upwind scheme used by the Cartesian FVCOM code can be directly used to calculate the advection terms in the momentum equations (see Chen et al. 2003a for details). Unlike the Cartesian version of FVCOM, the line integral for the flux calculation is done with respect to λ and φ rather than the arc length of the boundary line of the TCE or MCE. Thus, the meridian flux determined by v is calculated using $v\cos\varphi$, although the momentum equations are still solved for u and v. This method guarantees that the line integral flux calculation method is valid in the spherical coordinate system because the integral around a closed boundary path constructed by λ and φ equals zero.

The flux through the TCE is calculated using the line integral around the closed path defined by λ and φ. In the 2-D continuity equation, for example,

$$\iint\limits_{\Omega} \frac{\partial\zeta}{\partial t} r^2 \cos\varphi d\lambda d\varphi = \iint\limits_{\Omega} \frac{1}{r\cos\varphi} \{ \frac{\partial(\bar{u}D)}{\partial\lambda} + \frac{\partial[(\bar{v}\cos\varphi)D]}{\partial\varphi} \} r^2 \cos\varphi d\lambda d\varphi, \qquad (3.114)$$

Therefore, we have

$$\frac{\partial\zeta}{\partial t} = -\frac{r}{\Omega} [\oint\limits_{\varphi} (\bar{u}D)d\varphi' - \oint\limits_{\lambda} (\bar{v}\cos\varphi D)d\lambda'], \qquad (3.115)$$

where Ω is the area of the TCE. The line integral is done around the closed path of λ or φ, which guarantees volume conservation. The momentum and continuity equations for the 2-D mode are integrated numerically using the modified fourth-order Runge-Kutta

time-stepping scheme. A detailed description of this method was given in Chen et al. (2003a).

The unstructured triangular grid used in FVCOM allows us to nest the spherical and polar stereographic projection coordinates to avoid the singularity problem at the North Pole. Eight equilateral triangles are set up around the pole (Fig. 3.13), at which all variables at nodes and centroids of these triangles are calculated in the polar stereographic projection coordinates. Variables in MCEs and TCEs with no direct connection to the North Pole are calculated in spherical coordinates. The velocity in these two coordinates are converted to spherical coordinates by the relationship defined as

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \begin{pmatrix} -\sin\lambda & -\cos\lambda \\ \cos\lambda & -\sin\lambda \end{pmatrix} \begin{pmatrix} u_s \\ v_s \end{pmatrix}, \tag{3.116}$$
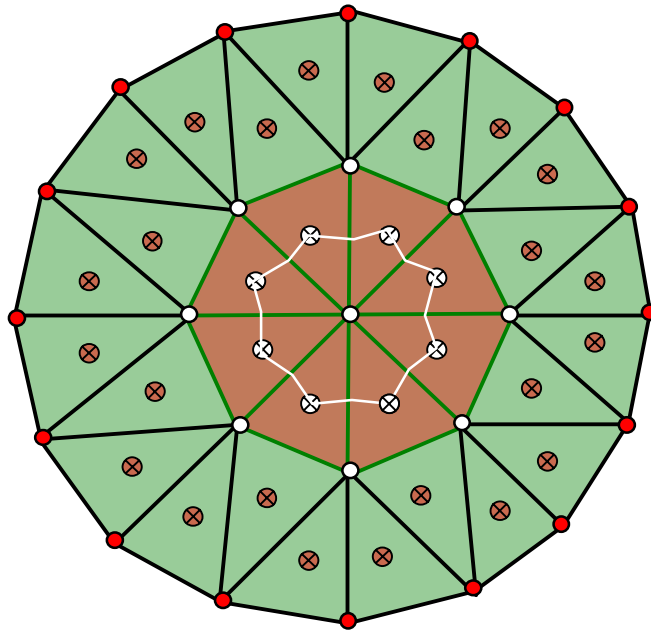
○   Node calculated using the polar stereographic projection coordinate.

⊗   Centroid calculated using the polar stereographic projection

●   Node calculated directly in the spherical coordinate system.

⊗   Centroid calculated directly in the spherical coordinate system.

Fig. 3.13: Illustration of the nested spherical-polar stereographic projection grid at the North Pole.

$$u_p = h_x \frac{dx}{dt}, \ v_p = h_y \frac{dy}{dt}, \ h_x = h_y = \frac{1 + \sin \phi}{2}. \qquad (3.117)$$

A so-called "North Pole Nesting Module" is used in the spherical-coordinate version of FVCOM, in which all cells connected to the North Pole are defined as the polar cells. This module allows an automatic selection of two systems based on the definition of a cell for the parallelized computation. This approach efficiently uses the flexibility of the unstructured grid to make FVCOM run directly in spherical coordinates for basin or larger (even global) scale ocean application without the need for "grid rotation" or "multi-projection" methods.

3.6 Ghost-Cell Treatment for the Coastal Boundary Condition

In the original version of FVCOM, the velocity in a cell adjacent to a solid boundary is updated using the same method as interior cells and is then adjusted so that the component normal to the wall is zero. This treatment works well for applications where the velocity field must only make small adjustments to the coastline, such as where the
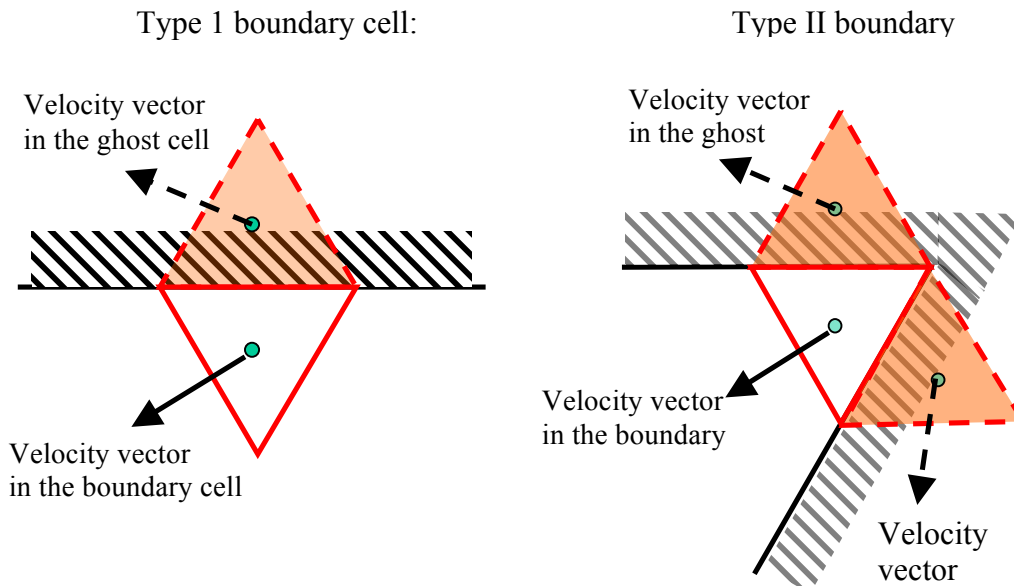
flow is predominantly tangential or the coast is smoothly varying. For cases where the coastal angle is rapidly changing or is normal to the locally forced flow, as in the case of an island, the original condition is unrealistic. By removing the normal component, an incoming wave is simply damped and energy reflection is reduced. We have thus

implemented an improved treatment in FVCOM using the traditional ghost cell approach in the current version. Brief descriptions of the methodology used in this treatment are given below.

For the type I solid boundary, where a triangle adjacent to a boundary has a single solid and two interior edges, a ghost cell is automatically created in the model in which the velocity has the same tangential component but opposite normal component to the adjacent boundary cell. The resulting velocity at the edge satisfies the proper solid boundary condition for inviscid flow. The adjacent boundary cell can thus be updated using the same techniques used in interior cells. For type II solid boundaries, where the boundary triangle has two solid edges, two ghost cells are automatically created in the model with the same treatment as the type I sold boundary cell. Although it does not affect the stability of the
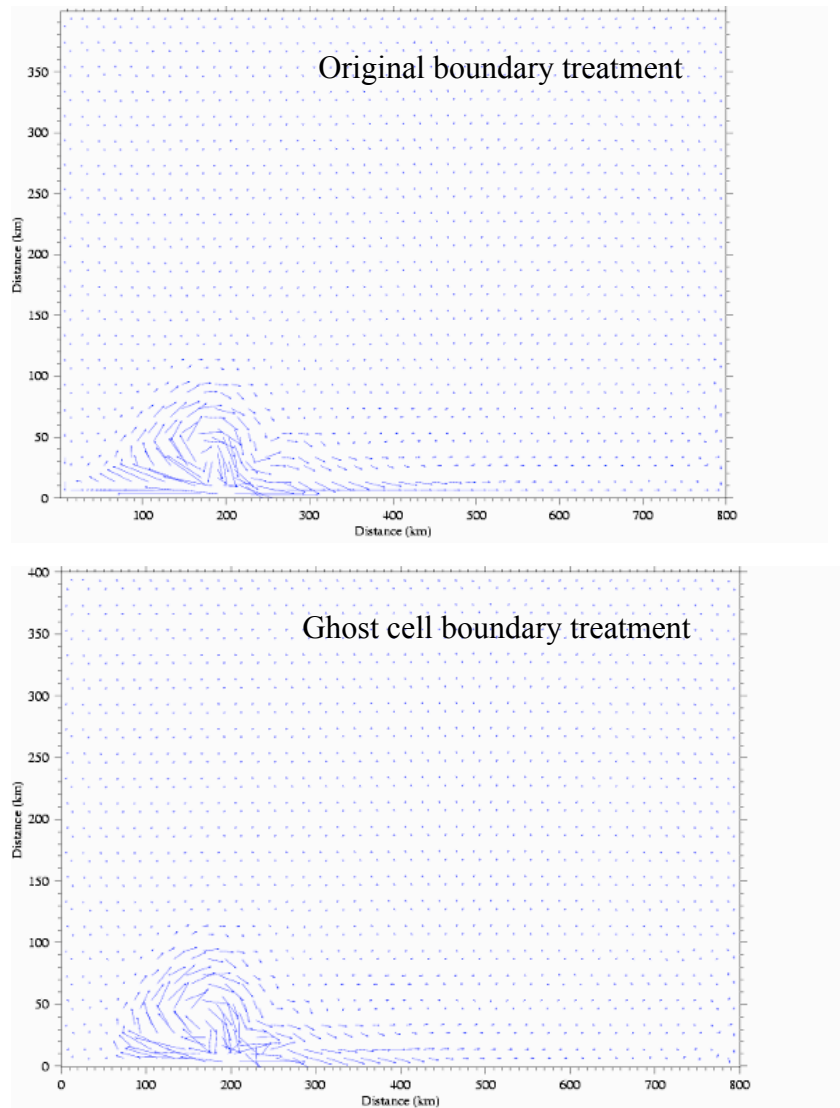
Fig. 3.14: The near-surface distribution of the current vector at the end of the 10th model day for a river discharge plume experiment.

model run, users should try to avoid the inclusion of type II cells when generating the mesh since they are not strongly linked to the interior flow.

Fig. 3.14 shows the model-computed distribution of the near-surface current vector at the end of the 10th model day for the case of a river discharge plume over an idealized continental shelf. It can be clearly seen that the original treatment of the solid boundary can affect the near-coastal distribution of the current and lead to an overestimation of the along-shelf water transport (as shown in the lower left corner).

The ghost cell treatment is particularly important for wave problems around an island. When the water is stratified, selection of the ghost cell boundary treatment might require a smaller time step when compared to the original setup. In the case where the boundary edge is normal to the dominant flow direction, the ghost cell boundary condition leads to a reflection of energy which can generate high frequency internal waves that require a reduction in time step.

Chapter 4: The Non-Hydrostatic FVCOM

4.1 Discrete Equations for Non-hydrostatic Pressure q

The details of the non-hydrostatic discretization algorithms were described in Lai's Ph.D. thesis (Lai, 2009) and also in two papers (Lai et al., 2010a,b). The benchmark test problems presented in Lai et al.'s papers were solved using the σ-coordinate system. Since then, Lai, Qi and Chen have upgraded the non-hydrostatic code to the generalized terrain-following coordinate system. A brief description is given here for the discretization forms of non-hydrostatic governing equations in the generalized terrain-following coordinate system.

The governing equations described in Chapter 2 are non-hydrostatic, in which the non-hydrostatic pressure q satisfies a Poisson equation that can be derived from the discrete decomposition using the fractional-step method (Chorin, 1968). Defining that u^*, v^* and w^* are the x, y and r components of the intermediate velocity, the eqs. (2.19)-(2.21) in Chapter 2 can be discretized as

$$\frac{u^*J^* - u^nJ^n}{\Delta t} = -F_x^n - \frac{a}{\rho_o}\left[\frac{\partial(qJ)^n}{\partial x} + \frac{\partial(qA_1)^n}{\partial r}\right] + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial u^*}{\partial r}\right) \tag{4.1}$$

$$\frac{v^*J^* - v^nJ^n}{\Delta t} = -F_y^n - \frac{a}{\rho_o}\left[\frac{\partial(qJ)^n}{\partial y} + \frac{\partial(qA_2)^n}{\partial r}\right] + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial v^*}{\partial r}\right) \tag{4.2}$$

$$\frac{w^*J^* - w^nJ^n}{\Delta t} = -F_z^n - \frac{a}{\rho_o}\frac{\partial q^n}{\partial r} + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial w^*}{\partial r}\right) \tag{4.3}$$

and

$$\frac{u^{n+1}J^{n+1} - u^*J^*}{\Delta t} = -\frac{1}{\rho_o}\left(\frac{\partial q'J}{\partial x} + \frac{\partial q'A_1}{\partial r}\right) \tag{4.4}$$

$$\frac{v^{n+1}J^{n+1} - v^*J^*}{\Delta t} = -\frac{1}{\rho_o}\left(\frac{\partial q'J}{\partial y} + \frac{\partial q'A_2}{\partial r}\right) \tag{4.5}$$

$$\frac{w^{n+1}J^{n+1} - w^*J^*}{\Delta t} = -\frac{1}{\rho_o}\frac{\partial q'}{\partial r} \tag{4.6}$$

where superscript n is an index presenting the nth time step; F_x and F_y represent the sum of advection, Coriolis, air pressure gradient, barotropic pressure gradient, and hydrostatic baroclinic pressure gradient, and horizontal diffusion terms in the x- and y-momentum equations (2.19)-(2.20) in Chapter 2; F_z is the sum of advection and horizontal diffusion

term in the *r*-momentum equation (2.21) in Chapter 2. q' is the perturbation non-hydrostatic pressure defined as

$$q^{n+1} = a \cdot q^n + q'$$ (4.7)

where a is a control flag: 0 for the projection method and 1 for the pressure correction method (Armfield and Street, 2002). The projection method first integrates the momentum equations under the hydrostatic pressure gradient condition to obtain the intermediate velocities and then corrects them to be divergent-free values using the non-hydrostatic pressure gradients (Mahadevan et al., 1996; Marshall et al., 1997; Casulli and Stelling, 1998; Kanarska and Maderich 2003; Heggelund et al., 2004). The pressure correction method also solves the momentum equations through two steps like the projection method except the non-hydrostatic pressure gradients are taken into the first step when the intermediate velocities are determined (Casulli, 1999; Stansby and Zhou, 1998; Zijlema and Stelling, 2005; Fringer et al., 2006; Kanarska et al., 2007).

Substituting the $(n+1)^{th}$ time step variables in equations (4.4)-(4.5) into the discretized continuity equation (2.22) in Chapter 2, we can derive the non-hydrostatic pressure Poisson equation for q' as

$$
\begin{aligned}
&\frac{\partial^2 q' J}{\partial x^2} + \frac{\partial^2 q' J}{\partial y^2} + \frac{\partial^2 q' A_1}{\partial x \partial r} + \frac{\partial^2 q' A_2}{\partial y \partial r} + \frac{\partial}{\partial r}(\frac{A_1}{J}\frac{\partial q' J}{\partial x}) + \frac{\partial}{\partial r}(\frac{A_2}{J}\frac{\partial q' J}{\partial y}) + \frac{\partial}{\partial r}(\frac{A_1}{J}\frac{\partial q' A_1}{\partial r}) \\
&+ \frac{\partial}{\partial r}(\frac{A_2}{J}\frac{\partial q' A_2}{\partial r}) + \frac{\partial}{\partial r}(\frac{1}{J}\frac{\partial q'}{\partial r}) = \frac{\rho_o}{\Delta t}(\frac{\partial u^* J}{\partial x} + \frac{\partial v^* J}{\partial y} + \frac{\partial u^* A_1}{\partial r} + \frac{\partial v^* A_2}{\partial r} + \frac{\partial w^*}{\partial r})
\end{aligned}
$$ (4.8)

Following the second-order approximate unstructured-grid finite-volume discretization methods of FVCOM, this equation is discretized by the second-order approximate finite-volume flux method as the same as in the hydrostatic version of FVCOM. It results in the linear system

$$Aq' = b$$ (4.9)

where A is a sparse coefficient matrix with a dimension of N_{node} (the node number) $\times K_{layer}$ (the vertical layer number) and b is the discrete array constructed from the right-hand side source terms in eq. (4.8). The resulting matrix is diagonally dominant and asymmetric, which can be efficiently solved by the Krylov subspace method (Saad, 2000).

The boundary conditions of q' are specified as follows. At the surface,

$$q' = 0;$$ (4.10)

at the bottom, no flux normal to the bottom of the slope results in

$$\frac{\partial q'}{\partial r} = \frac{J\tan\alpha}{(1+\tan^2\alpha)}\frac{\partial q'}{\partial \hat{n}}$$ (4.11)

where $\hat{n}$ is the unit horizontal directional vector component normal to the slope on the vertical grid layer surface and α is the slope of the bottom bathymetry. On the lateral solid boundary, the no-flux condition normal to the wall is specified, i.e.,

$$\frac{\partial q'J}{\partial n_h} = -(n_x \cdot \frac{\partial q'A_1}{\partial r} + n_y \cdot \frac{\partial q'A_2}{\partial r})$$ (4.12)

where n_h is the unit directional vector normal to the wall; n_x and n_y are the x and y components of n_h. At open boundaries, we assume that the flow approaches to the hydrostatic condition by assuming the RHS in eq. (4.8) is equal to zero or saying the intermediate velocities are the same as the final $(n+1)^{th}$ time step velocities. According to equations (4.4)-(4.5), a form similar to the no-flux condition (4.12) can be derived for the non-hydrostatic pressure gradient normal to the boundary lines on an open boundary. Such a treatment can effectively avoid the sharp gradient of non-hydrostatic pressure at the open boundary by directly setting $q' = 0$.

The discrete form of the Poisson eq. (4.8) is described as follows. The 5th and 6th terms in eq. (4.8) can be rewritten as

$$\frac{A_1}{J}\frac{\partial q'J}{\partial x} = \frac{A_1}{J}\frac{\partial q'J}{\partial x} + q'J\frac{\partial}{\partial x}(\frac{A_1}{J}) - q'J\frac{\partial}{\partial x}(\frac{A_1}{J}) = \frac{\partial q'A_1}{\partial x} - q'J\frac{\partial}{\partial x}(\frac{A_1}{J})$$

$$\frac{A_2}{J}\frac{\partial q'J}{\partial y} = \frac{A_2}{J}\frac{\partial q'J}{\partial y} + q'J\frac{\partial}{\partial y}(\frac{A_2}{J}) - q'J\frac{\partial}{\partial y}(\frac{A_2}{J}) = \frac{\partial q'A_2}{\partial y} - q'J\frac{\partial}{\partial y}(\frac{A_2}{J})$$ (4.13)

Replacing the 5th and 6th terms in eq. (4.8) using (4.13), eq. (4.8) can be written in the form of

$$\frac{\partial^2 q'J}{\partial x^2} + \frac{\partial^2 q'J}{\partial y^2} + 2\frac{\partial^2 q'A_1}{\partial x\partial r} + 2\frac{\partial^2 q'A_2}{\partial y\partial r} - \frac{\partial}{\partial r}[q'J\frac{\partial}{\partial x}(\frac{A_1}{J})] - \frac{\partial}{\partial r}[q'J\frac{\partial}{\partial y}(\frac{A_2}{J})]$$

$$+ \frac{\partial}{\partial r}(\frac{A_1}{J}\frac{\partial q'A_1}{\partial r}) + \frac{\partial}{\partial r}(\frac{A_2}{J}\frac{\partial q'A_2}{\partial r}) + \frac{\partial}{\partial r}(\frac{1}{J}\frac{\partial q'}{\partial r})$$ (4.14)

$$= \frac{\rho_o}{\Delta t}(\frac{\partial u^*J}{\partial x} + \frac{\partial v^*J}{\partial y} + \frac{\partial u^*A_1}{\partial r} + \frac{\partial v^*A_2}{\partial r} + \frac{\partial w^*}{\partial r})$$

where $A_1 = J\partial r/\partial x$ and $A_2 = J\partial r/\partial y$. Like all other tracer variables in FVCOM, q' is placed at the nodes of each triangle. This design can avoid the issues such as pressure-

velocity decoupling in collocated grid systems. Defining N (i=1, …,N) and M (j=1,… ,M) to be the total number of centroids and nodes in the computational domain, integrating eq. (4.14) over an individual control volume $\iint_{\Omega_j} \int_{r_{k+1}}^{r_k} (\quad) drdxdy$ yields

$$\oint_{\Omega_j}[\Gamma_k(\frac{\partial q'J}{\partial x})dy - \Gamma_k(\frac{\partial q'J}{\partial y})dx] + 2\oint_{\Omega_j}[\Delta_k(q'A_1)dy - \Delta_k(q'A_2)dx]$$

$$+\Delta_k\{\overline{q'\delta z}[\oint_{\Omega_j}(-\frac{A_1}{\delta z}dy + \frac{A_2}{\delta z}dx)]\} + \Omega_j[\Delta_k(\overline{\frac{A_1}{J}\frac{\partial q'A_1}{\partial r}})$$

$$+\Delta_k(\overline{\frac{A_2}{J}\frac{\partial q'A_2}{\partial r}})] + \Omega_j\Delta_k(\overline{\frac{1}{J}\frac{\partial q'}{\partial r}}) \tag{4.15}$$

$$=\frac{\rho_o}{\Delta t}[\oint_{\Omega_j}(\Gamma_k(u^*J)dy - \Gamma_k(v^*J)dx) + \Omega_j\Delta_k(\overline{u^*A_1}) + \Omega_j\Delta_k(\overline{v^*A_2}) + \Omega_j\Delta_k(\overline{w^*})]$$

where Ω_j is the area of the control volume at the center of the jth node; k is an index of the vertical r-layer varying from 1 to kb; δz is the layer thickness in the discrete form of $J = \delta z / \delta r$; Δ_k is an operator defined as the difference of a variable at k^{th} and $(k+1)^{th}$ r-layers; Γ_k is an operator defined as the integration of a variable over k^{th} and $(k+1)^{th}$ r-layers and the superscript "——" represents the average over the area Ω_j. A_1 and A_2 are functions of $D = \zeta + H$ that are known at the $(n+1)^{th}$ time step after the intermediate surface elevation is determined.

The discrete forms of each term on the left- and right-hand sides (LHS and RHS) of eq. (4.15) are described as follow. For the first term on LHS, with

$$\Delta x_1 = \frac{(x_{i,3} - x_{i,2})\delta z_{i,k+1/2}(1)}{2\Omega_i}; \ \Delta x_2 = \frac{(x_{i,1} - x_{i,3})\delta z_{i,k+1/2}(2)}{2\Omega_i}; \ \Delta x_3 = \frac{(x_{i,2} - x_{i,1})\delta z_{i,k+1/2}(3)}{2\Omega_i};$$

$$\Delta y_1 = \frac{(y_{i,3} - y_{i,2})\delta z_{i,k+1/2}(1)}{2\Omega_i}; \ \Delta y_2 = \frac{(y_{i,1} - y_{i,3})\delta z_{i,k+1/2}(2)}{2\Omega_i}; \ \Delta y_3 = \frac{(y_{i,2} - y_{i,1})\delta z_{i,k+1/2}(3)}{2\Omega_i},$$

then the first LHS term in eq. (4.15) can be rewritten as

$$\Gamma_k(\frac{\partial q'J}{\partial x}) = \Delta y_1 q'_{i,k+1/2}(1) + \Delta y_2 q'_{i,k+1/2}(2) + \Delta y_3 q'_{i,k+1/2}(3);$$

$$\Gamma_k(\frac{\partial q'J}{\partial y}) = \Delta x_1 q'_{i,k+1/2}(1) + \Delta x_2 q'_{i,k+1/2}(2) + \Delta x_3 q'_{i,k+1/2}(3),$$

where the subscript i indicates the ith centroid, $k+1/2$ is the mid-layer of the kth and $(k+1)^{th}$ r-layers; 1, 2 and 3 are the index of the three nodes of the ith triangle counted clockwise; $\delta z_{i,k+1/2}$ is the layer thickness between the kth and $(k+1)^{th}$ r-layers; and Ω_i is the area of the ith triangle. Therefore,

$$\oint_{\Omega_j}(\Gamma_k(\frac{\partial q'J}{\partial x})dy - \Gamma_k(\frac{\partial q'J}{\partial y})dx)$$

$$= \sum_{i=1}^{NE}\{\Gamma_k(\frac{\partial q'J}{\partial x})[dy_i(1)+dy_i(2)] - \Gamma_k(\frac{\partial q'J}{\partial y})[dx_i(1)+dx_i(2)]\}$$

(4.16)

where NE is the total number of triangles contained in the jth control volume with a center at the node point j, $[dx_i(1),dy_i(1)]$ and $[dx_i(2),dy_i(2)]$ represent the x and y lengths of the edge through the ith centroid for the jth control volume.

For the second term on LHS, defining that $q'_{i,k} = [q'_{i,k}(1)+q'_{i,k}(2)+q'_{i,k}(3)]/3$ (in which 1, 2 and 3 are the indexes of the three nodes of the ith triangle), we have

$$2\oint_{\Omega_j}[\Delta_k(q'A_1)dy - \Delta_k(q'A_2)dx] = 2\sum_{i=1}^{NE}(A_{1(i,k)}q'_{i,k} - A_{1(i,k+1)}q'_{i,k+1})[dy_i(1)+dy_i(2)]$$

$$-2\sum_{i=1}^{NE}(A_{2(i,k)}q'_{i,k} - A_{2(i,k+1)}q'_{i,k+1})[dx_i(1)+dx_i(2)]$$

(4.17)

where the subscript j is the jth node that is the center point of the jth control volume.

For the third term on LHS, defining that

$$\delta z_{j,k} = (\delta z_{j,k-1/2} + \delta z_{j,k+1/2})/2 \text{ and } \delta z_{j,k+1} = (\delta z_{j,k+1/2} + \delta z_{j,k+3/2})/2,$$

we have

$$\Delta_k\{\overline{q'\delta z}[\oint_{\Omega_j}(-\frac{A_1}{\delta z}dy + \frac{A_2}{\delta z}dx)]\} =$$

$$\{q'_{j,k}\delta z_{j,k}\sum_{i=1}^{NE}[-\frac{A_{1(i,k)}}{\delta z_{i,k}}(dy_i(1)+dy_i(2)) + \frac{A_{2i,k)}}{\delta z_{i,k}}(dx_i(1)+dx_i(2))]\}$$

(4.18)

$$-\{q'_{j,k+1}\delta z_{j,k+1}\sum_{i=1}^{NE}[-\frac{A_{1(i,k+1)}}{\delta z_{i,k+1}}(dy_i(1)+dy_i(2)) + \frac{A_{2i,k+1)}}{\delta z_{i,k+1}}(dx_i(1)+dx_i(2))]\}$$

For the fourth term on LHS, we have

$$\Omega_j[\Delta_k(\overline{\frac{A_1}{J}\frac{\partial q'A_1}{\partial r}}) + \Delta_k(\overline{\frac{A_2}{J}\frac{\partial q'A_2}{\partial r}})]$$

$$= \Omega_j[\frac{\overline{A_{1(j,k)}}\,\overline{A_{1(j,k-1/2)}} + \overline{A_{2(j,k)}}\,\overline{A_{2(j,k-1/2)}}}{\delta z_{j,k}}]q'_{j,k-1/2}$$

$$+\Omega_j[-\frac{\overline{A_{1(j,k)}}\,\overline{A_{1(j,k+1/2)}} + \overline{A_{2(j,k)}}\,\overline{A_{2(j,k+1/2)}}}{\delta z_{j,k}} - \frac{\overline{A_{1(j,k+1)}}\,\overline{A_{1(j,k+1/2)}} + \overline{A_{2(j,k+1)}}\,\overline{A_{2(j,k+1/2)}}}{\delta z_{j,k+1}}]q'_{j,k+1/2} \tag{4.19}$$

$$+\Omega_j[\frac{\overline{A_{1(j,k+1)}}\,\overline{A_{1(j,k+3/2)}} + \overline{A_{2(j,k+1)}}\,\overline{A_{2(j,k+3/2)}}}{\delta z_{j,k+1}}]q'_{j,k+3/2}$$

For the fifth term on LHS, we have

$$\Omega_j\Delta_k(\overline{\frac{1}{J}\frac{\partial q'}{\partial r}}) = \frac{\Omega_j}{\delta z_{j,k}}q'_{j,k-1/2} + [-\frac{\Omega_j}{\delta z_{j,k}} - \frac{\Omega_j}{\delta z_{j,k+1}}]q'_{j,k+1/2} + \frac{\Omega_j}{\delta z_{j,k+1}}q'_{j,k+3/2} \tag{4.20}$$

For the first term on the RHS of eq. (4.15), we have

$$\oint_{\Omega_j}(\Gamma_k(u^*J)dy - \Gamma_k(v^*J)dx)$$

$$= \sum_{i=1}^{NE}\{u^*_{i,k+1/2}\delta z_{i,k+1/2}[dy_i(1) + dy_i(2)] - v^*_{i,k+1/2}\delta z_{i,k+1/2}[dx_1(1) + dx_i(2)]\} \tag{4.21}$$

For the second term on RHS, we have

$$\Omega_j\Delta_k(\overline{u^*A_1}) + \Omega_j\Delta_k(\overline{v^*A_2})$$

$$= \Omega_j[\frac{(\overline{u^*}_{j,k-1/2}\overline{A_{1(j,k-1/2)}} + \overline{v^*}_{j,k-1/2}\overline{A_{2(j,k-1/2)}})\delta z_{j,k+1/2}}{\delta z_{j,k}}]$$

$$+\Omega_j\{(\frac{\delta z_{j,k-1/2}}{\delta z_{j,k}} - \frac{\delta z_{j,k+3/2}}{\delta z_{j,k+1}})[\overline{u^*}_{j,k+1/2}\overline{A_{1(j,k+1/2)}} + \overline{v^*}_{j,k+1/2}\overline{A_{2(j,k+1/2)}}]\} \tag{4.22}$$

$$-\Omega_j[\frac{(\overline{u^*}_{j,k+3/2}\overline{A_{1(j,k+3/2)}} + \overline{v^*}_{j,k+3/2}\overline{A_{2(j,k+3/2)}})\delta z_{j,k+1/2}}{\delta z_{j,k+1}}]$$

For the third term on RHS, we have

$$\Omega_j\Delta_k(\overline{w^*}) = \Omega_j(w^*_{j,k} - w^*_{j,k+1}) \ . \tag{4.23}$$

Substituting the discrete expressions (4.16)-(4.23) into eq. (4.15) produces a set of linear matrix equations (4.9). Three efforts have been made to solve eq. (4.9) with maximum efficiency. First, FVCOM-NH is parallelized under the MPI framework, as is the hydrostatic version of FVCOM, to efficiently use the workload balance feature of multi-processors (Cowles, 2008). Second, a scalable sparse matrix solver library (called

PETSc) (Balay et al., 2007) is implemented into the code to support the parallel computing environment for matrix solvers. Third, the high performance pre-conditional HYPRE software library (Falgout et al., 2002) is used to transform matrix *A* into a pre-conditioned matrix, which improves the efficiency using PETSc by a factor of three.

4.2 Mode-split and Semi-implicit Solvers

The non-hydrostatic FVCOM is coded with options for using mode-split and semi-implicit time integration methods. A brief description of these two methods is given below.

I) Mode-Split Method

The mode-split governing equations consist of the external (vertically averaged) and internal (3-D) modes. The mode-split time integration for the non-hydrostatic FVCOM follow the same procedure as that used in the hydrostatic FVCOM except for the 2-D/3-D adjustment and addition of the pressure Poisson equation. The discrete method used to solve the non-hydrostatic pressure Poisson equation has been described above. Here we briefly describe the time integration procedure for the external mode.

The non-hydrostatic governing equations for the external mode are given as

$$\frac{\partial \zeta}{\partial t} + \frac{\partial (\bar{u}D)}{\partial x} + \frac{\partial (\bar{v}D)}{\partial y} = 0 \tag{4.24}$$

$$\frac{\partial \bar{u}D}{\partial t} + \frac{\partial \bar{u}^2 D}{\partial x} + \frac{\partial \overline{uv}D}{\partial y} - f\bar{v}D - D\bar{F}_u - G_x - \frac{\tau_{sx} - \tau_{bx}}{\rho_o}$$

$$= -gD\frac{\partial \zeta}{\partial x} - \frac{D}{\rho_o}\frac{\partial p_a}{\partial x} - \frac{g}{\rho_o}\int_{-1}^{0}\{J[\int_{r}^{0} J(\frac{\partial \rho}{\partial x} + \frac{\partial r}{\partial x}\frac{\partial \rho}{\partial r})dr']\}dr' - \frac{1}{\rho_o}\int_{-1}^{0}[(\frac{\partial qJ}{\partial x} + \frac{\partial qA_1}{\partial r})]dr' \tag{4.25}$$

$$\frac{\partial \bar{v}D}{\partial t} + \frac{\partial \overline{uv}D}{\partial x} + \frac{\partial \bar{v}^2 D}{\partial y} + f\bar{u}D - D\bar{F}_v - G_y - \frac{\tau_{sy} - \tau_{by}}{\rho_o}$$

$$= -gD\frac{\partial \zeta}{\partial y} - \frac{D}{\rho_o}\frac{\partial p_a}{\partial y} - \frac{g}{\rho_o}\int_{-1}^{0}\{J[\int_{r}^{0} J(\frac{\partial \rho}{\partial y} + \frac{\partial r}{\partial y}\frac{\partial \rho}{\partial r})dr']\}dr' - \frac{1}{\rho_o}\int_{-1}^{0}(\frac{\partial qJ}{\partial y} + \frac{\partial qA_2}{\partial r})dr' \tag{4.26}$$

where *D* is total water depth, the definition of G_x and G_y were given in Chen et al. (2003a, (τ_{sx},τ_{sy}) and (τ_{bx},τ_{by}) are the *x* and *y* components of surface wind and bottom stresses, respectively, and the overbar "—" denotes vertical integration.

   Eqs. (4.24)-(4.26) are solved using the modified fourth-order Runge-Kutta time-stepping scheme as described in Chapter 3. Over the four-stage integration from n to $n+1$, q is given by its value at the nth time step and remains unchanged in the external mode integration. In order to give a best estimate of the free surface under non-hydrostatic conditions, the vertical integrated q's gradient is always considered in eqs. (4.25)-(2.26) no matter which projection or pressure correction method is used. The external and internal mode consistency adjustment used in the non-hydrostatic FVCOM is treated differently from the procedure used in the hydrostatic FVCOM. Under the hydrostatic approximation, with shorter time step, u and v calculated by the external mode are more accurate than $1/D \int_{-H}^{\xi} u\,dz$ and $1/D \int_{-H}^{\xi} v\,dz$ calculated from the internal mode. In this case, the internal velocity is adjusted to the external velocity at each time step to ensure the mode-split consistency (Chen et al., 2003a). In the non-hydrostatic case, however, the external mode only provides the immediate surface elevation for the use in determining the surface boundary condition of w and total water depth at the immediate time step in the pressure Poisson equation. Due to the lack of variation of q during the external mode integration, the surface elevation and vertically integrated velocity calculated by the external mode at the $(n+1)^{th}$ time step contain errors, which should be corrected inversely through the 2-D/3-D adjustment after q and the non-hydrostatic 3-D velocity at the $(n+1)^{th}$ time step are calculated. The true surface elevation and $(\bar{u}, \bar{v})$ at the $(n+1)^{th}$ time step is determined inversely by the divergence-free velocities (u, v, w) under the fully non-hydrostatic condition as follows. Defining

$$\bar{u}^{n+1} = \int_{-1}^{0} u^{n+1}\,dr \quad \text{and} \quad \bar{v}^{n+1} = \int_{-1}^{0} v^{n+1}\,dr \tag{4.27}$$

substituting $\bar{u}^{n+1}$ and $\bar{v}^{n+1}$ into the vertical integrated continuity equation (4.24), we update ζ^{n+1} by solving a fully implicit discrete equation given as

$$\frac{\zeta^{n+1} - \zeta^{n}}{\Delta t} + \frac{\partial[\bar{u}^{n+1}(H + \zeta^{n+1})]}{\partial x} + \frac{\partial[\bar{v}^{n+1}(H + \zeta^{n+1})]}{\partial y} = 0 \tag{4.28}$$

Eq. (4.28) results in a 2-D asymmetric and diagonally dominant matrix with a stencil equal to the sum of the surrounding node points contained in a control volume, which can

be solved efficiently. This approach avoids the artificial adjustment that is required in the hydrostatic version of FVCOM or other time-split models. Because the true values of ζ^{n+1}, $\bar{u}^{n+1}$ and $\bar{v}^{n+1}$ are determined using the 3-D divergence free velocity under the fully non-hydrostatic conditions, the volume fluxes for external and external modes are matched exactly.

II) Semi-implicit Method

In the semi-implicit time-stepping method, the horizontal momentum equations (2.19)-(2.20) in Chapter 2 are written in a half-discretized form (Casulli and Cattani, 1994) given as

$$
\begin{aligned}
\frac{\partial uJ}{\partial t} + \frac{\partial u^2 J}{\partial x} + \frac{\partial uvJ}{\partial y} + \frac{\partial u\omega}{\partial r} - fvJ = -gJ[(1-\theta)\frac{\partial \zeta^n}{\partial x} + \theta\frac{\partial \zeta^{n+1}}{\partial x}] - \frac{J}{\rho_o}\frac{\partial p_a}{\partial x} \\
- \frac{gJ}{\rho_o}[\int_r^0 J(\frac{\partial \rho}{\partial x} + \frac{\partial r'}{\partial x^*}\frac{\partial \rho}{\partial r'})dr'] - \frac{1}{\rho_o}(\frac{\partial qJ}{\partial x} + \frac{\partial qA_1}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial u}{\partial r}) + JF_u
\end{aligned}
\tag{4.29}
$$

$$
\begin{aligned}
\frac{\partial vJ}{\partial t} + \frac{\partial uvJ}{\partial x} + \frac{\partial v^2 J}{\partial y} + \frac{\partial v\omega}{\partial r} + fuJ = -gJ[(1-\theta)\frac{\partial \zeta^n}{\partial y} + \theta\frac{\partial \zeta^{n+1}}{\partial y}] - \frac{J}{\rho_o}\frac{\partial p_a}{\partial y} \\
- \frac{gJ}{\rho_o}[\int_r^0 J(\frac{\partial \rho}{\partial y} + \frac{\partial r'}{\partial y^*}\frac{\partial \rho}{\partial r'})dr'] - \frac{1}{\rho_o}(\frac{\partial qJ}{\partial y} + \frac{\partial qA_2}{\partial r}) + \frac{\partial}{\partial r}(\frac{K_m}{J}\frac{\partial v}{\partial r}) + JF_v
\end{aligned}
\tag{4.30}
$$

and the transformed vertical velocity ω satisfies the semi-implicit form of the continuity equation defined as

$$
\frac{\partial J}{\partial t} + [(1-\theta)\frac{\partial (uJ)^n}{\partial x} + \theta\frac{\partial (uJ)^{n+1}}{\partial x}] + [(1-\theta)\frac{\partial (vJ)^n}{\partial y} + \theta\frac{\partial (vJ)^{n+1}}{\partial y}] + \frac{\partial \omega}{\partial r} = 0 .
\tag{4.31}
$$

Integrating eq. (4.31) from surface to bottom, we have

$$
\begin{aligned}
\frac{\partial \zeta}{\partial t} + \{\int_{-1}^0 [(1-\theta)\frac{\partial (uJ)^n}{\partial x}]dr' + \int_{-1}^0 [\theta\frac{\partial (uJ)^{n+1}}{\partial x}]dr'\} \\
+ \{\int_{-1}^0 [(1-\theta)\frac{\partial (vJ)^n}{\partial y}]dr' + \int_{-1}^0 [\theta\frac{\partial (vJ)^{n+1}}{\partial y}]dr'\} = 0
\end{aligned}
\tag{4.32}
$$

The eqs. (4.29)-(4.30) and (4.32) can be solved semi-implicitly by the following steps. Step I: at each time step, the momentum eqs. (4.29)-(4.30) are discretized in the form of the $(n+1)^{th}$ time step layered transports given as

$$\int_{r_{k+1}}^{r_k} (uJ)^{n+1} dr' = Flux_u^n - g\Delta t \cdot \int_{r_{k+1}}^{r_k} J\theta \frac{\partial \zeta^{n+1}}{\partial x} dr',$$

$$\int_{r_{k+1}}^{r_k} (vJ)^{n+1} dr' = Flux_v^n - g\Delta t \cdot \int_{r_{k+1}}^{r_k} J\theta \frac{\partial \zeta^{n+1}}{\partial y} dr'$$

(4.33)

where $Flux_u^n$ and $Flux_v^n$ include all the explicit calculated terms of the *x*- and *y*-momentum equations at the nth time step, respectively. Step II: substitute eq. (4.33) into the continuity equation (4.32) to produce a 2-D matrix for the intermediate free surface elevation (ζ^*). Step III: solve the matrix of ζ^* and then eq. (4.33) to determine ζ^*, u^* and v^*. Step IV: substitute ζ^*, u^* and v^* into eqs. (4.4)-(4.6) to determine the $(n+1)^{th}$ time divergent-free velocity field after solving the non-hydrostatic pressure *q*. Step V: correct the free surface elevation by solving a semi-implicit form of the continuity equation given as

$$\frac{\zeta^{n+1} - \zeta^n}{\Delta t} + (1-\theta)\frac{\partial[\bar{u}^n(H+\zeta^n)]}{\partial x} + \theta\frac{\partial[\bar{u}^{n+1}(H+\zeta^{n+1})]}{\partial x}$$
$$+ (1-\theta)\frac{\partial[\bar{v}^n(H+\zeta^n)]}{\partial y} + \theta\frac{\partial[\bar{v}^{n+1}(H+\zeta^{n+1})]}{\partial y} = 0$$

(4.34)

4.3 Model Setup of Non-hydrostatic FVCOM

The model setup is described in general in Chapter 18. Here we provide a brief description of the procedures required to run the non-hydrostatic FVCOM. Generally speaking, users first need to select the "NH" option in the *make.inc* and compile the code. One parameter called "PROJ_SWITCH" needs to be specified. This parameter allows users to choose either projection or pressure correction method. The pseudo code appears as :

```
        IF (current time step <= PROJ_SWITCH) THEN
            Implement projection method
        ELSE
            Implement pressure correction method
        ENDIF
```

If setting PROJ_SWITCH to be any integer number less than 0, the projection method will be selected. If setting PROJ_SWITCH greater than the maximum model integration

steps, the pressure correction method will be used. FVCOM includes an option to use these two methods in different time integration periods. By specifying PROJ_SWITCH time step, the projection method will be used from the beginning of the time integration to the specified time step and then it will switch to the pressure correction method after that. In some real-application cases, the initial inverted non-hydrostatic pressure field contains relative large errors. Such errors could be accumulated and lead to the failure of computation if the pressure correction method is applied from the beginning. The option described above will be helpful to build up a more correct non-hydrostatic pressure field with project method before turning on the pressure correction method.

4.4. Validations

The non-hydrostatic FVCOM has been well validated by four idealized benchmark test problems in homogeneous and stratified fluids. The results have been published in Lai et al. (2010a,b). The pdf versions of these two papers are available at the FVCOM website: http://fvcom.smast.umassd.edu/Extra/publication.html. A detailed description and discussion are given in Lai's Ph.D. dissertation (Lai, 2009). The pdf version of Lai' dissertation can be obtained by contacting Lai or Chen. Animations of the test results can be viewed and downloaded from the FVCOM website http://fvcom.smast.umassd.edu/research_projects/NH-FVCOM/index.html.

Here we select the lock exchange problem as an example for the non-hydrostatic model validation. Consider a rectangular tank filled with two fluids of different densities (hereafter referred as light and heavy) that are separated initially by a vertical gate at the tank center (Lai et al., 2010a). The tank has a length (L) of 0.8 m, a width (W) of 0.008 m and a static water depth (H) of 0.1 m. Assume that the density (ρ) is linearly proportional to salinity (S) given by

$$\rho = 999.972 \times (1 + 0.75 \times 10^{-3} S),$$

and ρ_1 = 999.972 kg/m^3 and ρ_2 = 1000.991371 kg/m^3. This gives a reduced gravity (g') as

$$g' = g\Delta\rho / \rho_o = 0.01 \text{ m/s}^2$$

where ρ_o is the reference density specified as 1000 kg/m^3 and g is the gravitational acceleration constant with a value of 9.81 m/s^2.

The non-hydrostatic FVCOM successfully reproduces the Kelvin-Helmholtz (KH) instability and gravity currents. The process of gravitational adjustment begins after the gate is removed at $t = 0$ (t is the time). The flow field is deformed from the initial state as the heavy fluid flows underneath the lighter fluid and the velocity across the interface is of opposite sign (Fig. 4.1). The KH instability appears at a time when the velocity shear between the two fluids is greater than the critical value of the restoring force determined by the density gradient. As a result of instability, a chain of well-defined vortices develops along the interface as the heads of the two fluids advance toward the end walls. Following the reflection at the end walls, the gravity-driven water flows reverse direction and a complex flow field characterized by overturning, strong mixing and vortices develop. The FVCOM-NH computed evolution of the gravity current and instability structures shown in Fig. 4.1 is in good agreement with laboratory experiment results described in previous literature.
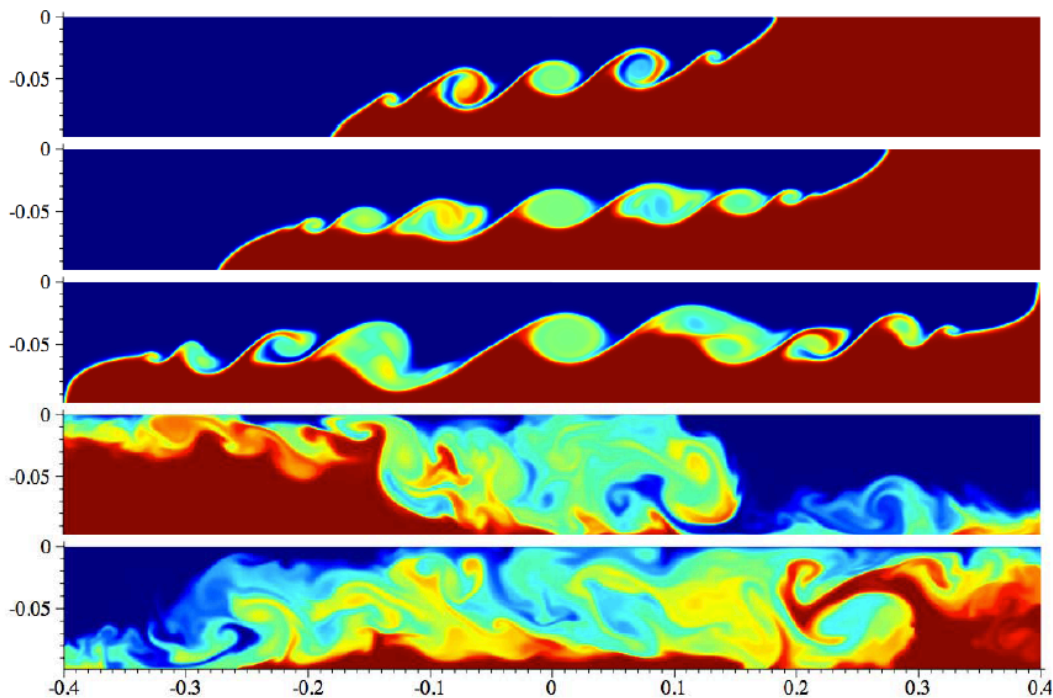
Figure 4.1 Density fields at 12, 18, 27, 78 and 144 (top to bottom) seconds after the vertical wall is removed at x = 0 for the inviscid non-hydrostatic FVCOM simulation.

Chapter 5: External Forcing

External physical forcings driving FVCOM include the surface wind stress, heat flux, precipitation/evaporation, tides, river discharges and groundwater flux. Brief descriptions of these forcing processes and how they are applied in FVCOM are given below.

4.1. Wind Stress, Heat Flux and Precipitation/Evaporation

The surface wind stress, heat flux and precipitation/evaporation are added to FVCOM using state-of-the-art methods that are widely used in ocean circulation modeling. Users can either run FVCOM with a constant uniform forcing or with time-dependent, spatially non-uniform forcing fields specified from observational data or from the output of a meteorological model or a combination of both. FVCOM has been loosely coupled with the fifth-generation NCAR/Penn State mesoscale meteorological model (called MM5) and more recently with the Weather Research Forecast (WRF) model, so that the output of the MM5/WRF wind stress, heat flux and precipitation/evaporation fields can be used to drive FVCOM, and the FVCOM assimilated surface water temperature can be feedback to the MM5/WRF model for use in the sensible and latent heat flux estimation (Chen et al., 2005).

4.2. Tidal Forcing

FVCOM has incorporated the Foreman (1978) tidal forecasting program to compute realistic tidal elevation data for the initial conditions and on the open boundary. Up to eight tidal harmonic constituents can be included in the model. They are given as

$$\zeta_o = \bar{\zeta}_o + \sum_{i=1}^{N_o} \hat{\zeta}_i \cos(\omega_i t - \theta_i) \tag{5.1}$$

where $\bar{\zeta}_o$ is the mean elevation relative to a water level at rest; $\hat{\zeta}_i$, ω_i and θ_i are the amplitude, frequency and phase of the ith tidal constituent. N_o is the total number of tidal constituents, which is set up to be eight in the current version of FVCOM. This number, however, can be increased according to users' needs. The eight included tidal constituents are: 1) S_2 tide (period = 12 hours); 2) M_2 tide (period = 12.42 hours); 3) N_2 tide (period = 12.66 hours); 4) K_2 (period = 11.97 hr); 5) K_1 tide (period = 23.94 hours); 6) P_1 tide

(period = 24.06 hours); 7) O_1 tide (period = 25.82 hours); and 8) Q_1 (period = 26.87 hours).

In larger scale regional applications, the equilibrium tide generated by the gravitational-centrifugal potential cannot be ignored and the potential gradient force must be included in the tidal simulation. The equilibrium tidal gradient force is derived from the elevations given as

$$\overline{\zeta}_{e(semi)} = \sum_{i=1}^{N_{semi}} \beta_i A_{e_i} \cos^2 \varphi \cos(\omega_{e_i} t + 2\lambda) \tag{5.2}$$

for semidiurnal tides and

$$\overline{\zeta}_{e(diurnal)} = \sum_{i=1}^{N_{diurnal}} \beta_i \hat{A}_{e_i} \cos 2\varphi \cos(\hat{\omega}_{e_i} t + \lambda) \tag{5.3}$$

for diurnal tides. Here A_{e_i} and ω_{e_i} are the amplitude and frequency of the ith semidiurnal equilibrium tidal elevation, respectively; φ is latitude; λ is longitude; N_{semi} and $N_{diurnal}$ are the total number of the semidiurnal and diurnal equilibrium tidal constituents included in the model, respectively, and β_i is the parameter specified as

$$\beta_i = 1 + K_{Love} - H_{Love} . \tag{5.4}$$

K_{Love} and H_{Love} are different for different tidal constituents. In the current version of FVCOM, eight equilibrium tidal constituents are included and the values of β_i for these constituents are given in Table 5.1.

Table 5.1: Parameters of K_{Love} and H_{Love}

	S_2	M_2	N_2	K_2	K_1	P_1	O_1	Q_1
β_i	0.693	0.693	0.693	0.693	0.736	0.706	0.695	0.695

Similarly, FVCOM also includes the atmospheric tidal potential forcing. For example, the atmospheric S_2 tidal forcing is derived from the elevation given as

$$\overline{\zeta}_{a(S_2)} = \overline{A}_{a(S_2)} \cos^2 \varphi \cos(\omega_{a(S_2)} t + 2\lambda - \alpha_a) \tag{5.5}$$

where $\overline{A}_{a(S_2)}$ and $\omega_{a(S_2)}$ are the amplitude and frequency of the atmospheric S_2 tidal elevation, and α_a is a parameter given as 112.0. $\overline{A}_{a(S_2)}$ and $\omega_{a(S_2)}$ are specified according to Haurwitz (1956) (see Chapman and Lindzen, 1970).

The tidal forcing at the open boundary can be specified by either amplitude/phase or by time series calculated using Foreman's or other tidal forecast models. In the former case, the model time base can be arbitrary, i.e., not tied to a specific Gregorian time. The harmonic analysis is carried out by using a least square fitting. In the second case, true clock time must be specified and the results can be analyzed directly using Foreman's harmonic analysis program. Normally amplitude/phase specification is used to test the model, and the prescribed elevation is used in simulations for a specific Gregorian time period.

5.3. Methods to Add the Discharge from the Coast or a River

FVCOM incorporates two methods for including the discharge of fresh water or tracer transport from the coastal solid boundary. The first is to inject the water into the tracer control element (TCE) and the second is to input the water into the momentum control element (MCE). In each of these methods, the tracer concentration (such as salinity, temperature or others) can be either specified or calculated through the tracer equation. The discrete expressions for these two approaches are described in detail below.

5.3.1. The TCE Method

Let Q be the water volume transport into a TCE with an area of Ω^{ξ} and a depth of D (shown in Fig. 5.1). The surface elevation at the coastal node in this TCE can be calculated by

$$\frac{\partial \zeta}{\partial t} = [-\oint_s v_n D ds + Q] / \Omega^{\xi} , \qquad (5.6)$$

where v_n is the velocity component normal to the boundary line of the TCE and s is the closed trajectory of the boundary of the TCE. The way to include Q in the continuity equation is equivalent to adding the flux into a TCE from its coastal boundary lines (see

the heavy line shown in Fig. 5.1). Since this boundary line links to the two momentum control elements (MCE) (shaded gray in Fig. 5.1), the contribution of Q to the momentum in these two elements needs to be taken into account.
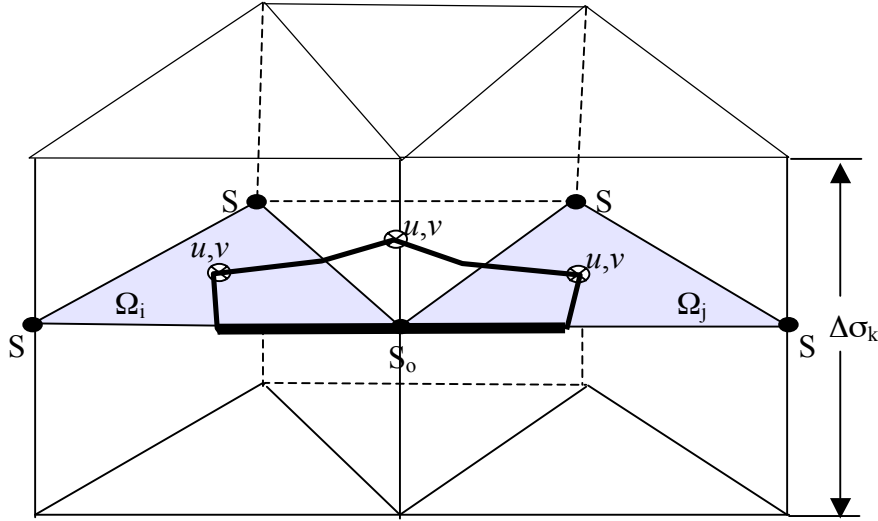
Fig. 5.1: Illustration of the river discharge as a point.

For the external mode, defining that l_i and l_j are half-lengths of the coastal sides of the two MCE triangles with areas Ω_i and Ω_j, respectively, the vertically-averaged x and y components of the velocity resulting from Q equal

$$U_o = \frac{Q\cos\hat{\theta}}{D(l_i + l_j)}, \; V_o = \frac{Q\sin\hat{\theta}}{D(l_i + l_j)} \tag{5.7}$$

where $\hat{\theta}$ is the angle of the coastline relative to the x direction. The contributions of Q to the x and y vertically-integrated momentum equations in the MCE with an area of Ω_i or Ω_j are given as $0.5QU_o$ and $0.5QV_o$, respectively.

For the internal mode, define that R_{Qk} is the percentage of Q in the kth sigma layer which satisfies the condition

$$\sum_{k=1}^{KM-1} R_{Qk} = 1, \tag{5.8}$$

where KM is the number of sigma levels in the vertical. The transport entering the kth sigma layer in the TCE is equal to QR_{Qk} and the x and y components of the velocity resulting from this amount of water transport are given as

$$U_{ok} = \frac{QR_{Qk}\cos\hat{\theta}}{D(l_i + l_j)\Delta\sigma_k}; V_{ok} = \frac{QR_{Qk}\sin\hat{\theta}}{D(l_i + l_j)\Delta\sigma_k} \tag{5.9}$$

where $\Delta\sigma_k$ is the thickness of the kth sigma layer. Therefore, the contributions of QR_{Qk} to the x and y momentum equation in the kth sigma layer of the MCE with an area Ω_i or Ω_j are given as $0.5QR_{Qk}U_{ok}$ and $0.5QR_{Qk}V_{ok}$, respectively.

The tracer concentration (such as salinity, temperature, or others) at the coastal node of the TCE can be either specified or calculated. For the first case, the tracer concentration at the coastal node is specified by users at each time step, so that no calculation is needed to solve the tracer equation for the TCE where Q is added. This method is built on an assumption that no mixing occurs in the TCE where the water is injected from the coast or rivers. It is also the method that is usually used in finite-difference models for point sources. The advantage of this method is that it is conceptually simple, but it may cause unrealistic buoyancy gradients near the discharge source, especially for the case with coarse horizontal resolution.

For the second case, the tracer concentration at the coastal node where Q is added is calculated directly from the tracer equation, with an assumption that the water injected into the system directly contributes to the tracer transport in the TCE (where the discharge source is located) and the tracer concentration at the coastal node of this TCE is determined by the adjusted net tracer flux and mixing. For example, defining that S_{ok} is the salinity in the kth sigma layer at the coastal node of the TCE where Q is added, it can be determined by the salinity equation in an integral form as

$$\frac{\partial S_{ok}D}{\partial t} = [-\oint_{s-(l_i+l_j)} v_{nk}S_k Dds + \iint_{\Omega^\xi} F_s dxdy + QR_{Qk}\hat{S}_{ok}]/\Omega^\xi \tag{5.10}$$

where v_{nk} is the velocity component normal to the boundary of the TCE in the kth sigma layer, S_k is the salinity at nodes of triangles connecting to the coastal node of the TCE, F_s is the horizontal and vertical diffusion terms in the salinity equation, and $\hat{S}_{ok}$ is the salinity contained in the water volume Q.

5.3.2. The MCE Method

Let Q be the water volume transport into a MCE, $\hat{\theta}$ is the angle of the coastline relative to the x direction, and l is the length of the coastal boundary of the MCE (Fig. 5.2). The vertically-averaged x and y components of the velocity driven by Q can be estimated as

$$U_o = \frac{Q}{Dl}\cos\hat{\theta} \; ; \; V_o = \frac{Q}{Dl}\sin\hat{\theta} \, . \tag{5.11}$$

Using the same definition of R_{Qk} described in (5.8), the x and y components of the velocity in the kth sigma layer in the MCE are given as
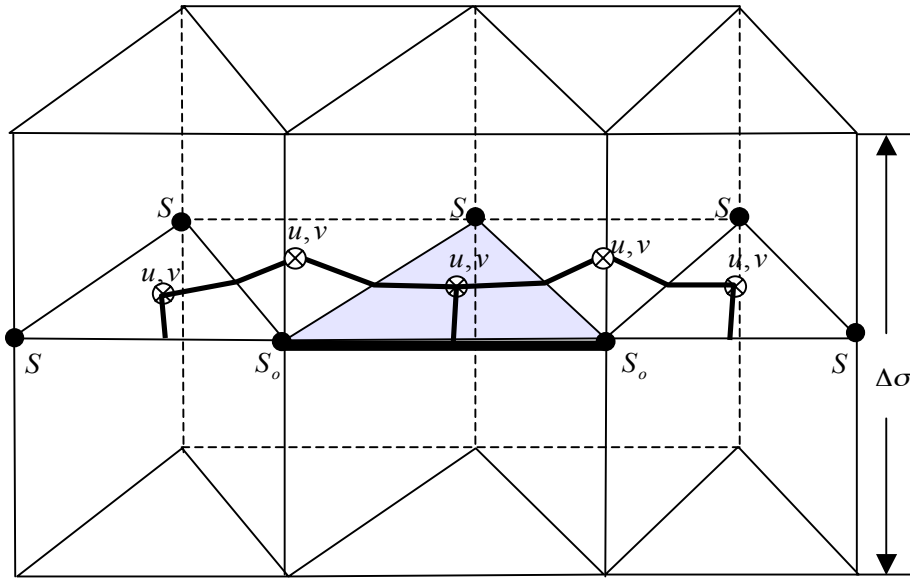
Fig. 5.2: Illustration of the freshwater discharge entering a MCE from the coastal boundary.

$$U_{ok} = \frac{QR_{Qk}\cos\hat{\theta}}{Dl\Delta\sigma_k} \; ; \; V_{ok} = \frac{QR_{Qk}\sin\hat{\theta}}{Dl\Delta\sigma_k} \quad . \tag{5.12}$$

Therefore, the contributions of the freshwater discharge to the external x and y momentum equations of the MCE are QU_o and QV_o, respectively, while the contributions to the internal x and y momentum equations of the MCE in the kth sigma layer are equal to $QR_{Qk}U_{ok}$ and $QR_{Qk}V_{ok}$, respectively.

Because the freshwater discharge is injected in the computational domain from a single MCE, we assume that there is an along-coast gradient of sea level due to this discharge. A simple way to satisfy this condition is to choose the same geometric shape for the two surrounding TCEs that connect to the MCE where the freshwater is input and assume that the freshwater equally flow into these two TCEs. Let i and j represent the two TCEs connected to the freshwater source, then

$$Q_i = Q_j = Q/2 . \tag{5.13}$$

Note that this assumption is only valid when the two surrounding TCEs have the same geometric shapes. Therefore, the surface elevation at the coastal node in the ith and j^{th} TCEs can be calculated by

$$\frac{\partial \zeta_I}{\partial t} = [-\oint_s v_{nI} D_I ds + Q_I]/\Omega_I^\xi , \tag{5.14}$$

where I are either i or j, and v_{nI} is the velocity component normal to the boundary of the Ith TCE, and s is the closed trajectory of the boundary of this TCE.

The tracer concentration (such as salinity, temperature, or others) at the coastal node of the ith or j^{th} TCE can also be either specified or calculated. The method used to include the freshwater discharge into the tracer equation is the same as those described in the TCE method.

5.4. Criterions for Horizontal Resolution and Time Step

In most coastal ocean models, the tracer concentration at grid points representing freshwater runoff is specified. This approach seems very straightforward because the tracer concentration of the runoff is easily determined by direct measurements taken at either the mouth or upstream of a river. In view of numerical computation, however, this

method is built on an assumption that no mixing occurs in the individual computational volume connected to the freshwater source. It should be generally sound in the case with sufficient horizontal resolution, but it may cause unrealistic buoyancy gradients near the source of the runoff and thus exaggerate the spatial scale and propagation speed of the low-salinity plume in the case with coarse horizontal resolution. A criterion is derived here to find a minimum horizontal resolution or time step to ensure the mass conservation in the calculation of a tracer concentration that is injected from a point source.

Considering a case with no vertical and horizontal diffusion, the vertical integral form of the salinity equation in FVCOM with river runoff is given as

$$\frac{\partial SD}{\partial t} = [-\oint v_n SD ds + Q\hat{S}_o]/\Omega^\xi \tag{5.15}$$

where S is the salinity at nodes of triangles connecting to the coastal node of the TCE, v_n is the velocity component normal to the boundary of the TCE, Q is the volume transport of the runoff, $\hat{S}_o$ is the salinity of the runoff water, and Ω^ξ is the area of the TCE containing the runoff. For simplification, let us consider a first-order forward time integration scheme. Assuming that at the nth time step, the computational domain is filled with water with salinity S^n, then at the $(n+1)^{\text{th}}$ time step, the salinity at the node point connected to the runoff can be estimated by

$$S^{n+1} = S^n \left(\frac{D^n}{D^{n+1}} - \frac{\Delta t \oint v_n D^n ds}{D^{n+1}\Omega^\xi}\right) + \frac{\Delta t Q \hat{S}_o}{D^{n+1}\Omega^\xi} \quad , \tag{5.16}$$

where Δt is the internal mode time step used for numerical computation. For simplification, a forward numerical scheme is used here for time integration. Replacing the transform term in (5.16) using the continuity equation yields

$$S^{n+1} = S^n + \frac{\Delta t Q}{D^{n+1}\Omega^\xi}(\hat{S}_o - S^n) \quad . \tag{5.17}$$

Specifying that $\hat{S}_o = 0$ for a freshwater discharge case, (5.17) can be simplified as

$$S^{n+1} = (1 - \frac{\Delta t Q}{D^{n+1}\Omega^\xi})S^n \quad . \tag{5.18}$$

To ensure positivity of the tracer, (5.18) must satisfy

$$1 - \frac{\Delta t Q}{D^{n+1} \Omega^{\zeta}} \geq 0 \quad \text{or} \quad D^{n+1} \geq \frac{\Delta t Q}{\Omega^{\zeta}} \quad . \tag{5.19}$$

Assuming that $\Omega^{\zeta} \sim L^2$, $\Delta t \sim I_{\text{split}} L / \sqrt{gD}$, $D^{n+1} \sim D$, where L and D are the typical scales of the horizontal resolution of the TCE and water depth, then in order to ensure that the salinity at the node point connected to the runoff will remain zero, L must scale as

$$L \sim \frac{I_{\text{split}} Q}{D \sqrt{gD}} \quad . \tag{5.20}$$

For given values of $Q \sim 10^3 \, \text{m}^3/\text{s}$, $D \sim 10$ m, and $I_{split} = 10$, for example, L should be of order 100 m.

In many coastal ocean numerical experiments, computer limitations frequently force L to be set at values significantly larger than that required in the criterion (5.20). This suggests that if we consider the mass conservation of the TCE connected to the runoff, the salinity calculated by (5.17) would be smaller than the salinity specified for the runoff water. On the other hand, if the horizontal resolution specified in the model is coarser than that required in (5.20), then the model would not ensure mass conservation in the TCE that is connected to the runoff and would also exaggerate the salinity gradient near the runoff source. Subsequently, it would overestimate the intensity of the low-salinity plume and cause an unrealistically faster propagation of the plume in the downstream direction.

In addition, for given local water depth, discharge rate, and horizontal resolution, the internal time step must satisfy the following criterion in order to ensure mass conservation:

$$\Delta t \leq \frac{D^{n+1} \Omega^{\zeta}}{Q} \tag{5.21}$$

The time step must be reduced as the freshwater discharge rate increases. This criterion must be satisfied at the shallow river mouth with a large discharge rate. Reducing the

horizontal resolution is also helpful, but in the case of a very large river discharge rate, (5.21) should be checked when an internal time step is specified.

This diagnostic analysis described above is made for a simple first-order discrete scheme, which differs from the discrete scheme used in FVCOM. The required condition shown in the criterion (5.20) might be less restrictive when the higher order discrete scheme is used. Recently we added the FTC method into the second-order upwind scheme, which is helpful to avoid the occurrence of the negative salinity value at the mouth of the river when the horizontal resolution is coarse.

5.5. Groundwater Input through the Bottom

Two methods for the groundwater input have been implemented in FVCOM. The first only considers the change of the salinity due to the groundwater input and no volume flux of the groundwater is added. The second includes both salt and volume transports at the bottom and also the change of the volume in the continuity equation. Brief descriptions for these two methods are given below.

5.5.1. A Simple Salt Balance Groundwater Flux Form

Assuming that the volume flux of the groundwater is an order of magnitude smaller than the total volume of the water body, we can ignore the change of the volume due to the groundwater input and only treat the groundwater input through a salt balance model. Let Q_b (units: m^3/s) be the freshwater discharge rate at the bottom, V_b be the volume of the bottom TCE with the freshwater discharge from the bottom, and Ω is the bottom area of that TCE. Our assumption is that the change of the salinity in this TCE due to freshwater input is equal to the salinity loss due to the bottom diffusion. That is

$$\frac{V_b \times S}{V_b + Q_b \times \Delta t} = \frac{V_b \times S - (K_h \frac{\partial S}{\partial z})\Big|_{z=-H} \times \Omega \times \Delta t}{V_b} \tag{5.22}$$

where K_h is the vertical salinity diffusion coefficient (m^2/s). Considering a salinity loss over a unit time interval ($\Delta t = 1$), (5.22) can be re-written as

$$K_h \frac{\partial S}{\partial z}\bigg|_{z=-H} = \frac{V_b \times Q_b \times S}{(V_b + Q_b) \times \Omega}$$ (5.23)

In the sigma coordinate, (5.23) is rewritten as

$$\frac{\partial S}{\partial \sigma}\bigg|_{\sigma=-1} = \frac{D \times V_b \times Q_b \times S}{K_h(V_b + Q_b) \times \Omega} .$$ (5.24)

It should be pointed out here that this condition works only for the groundwater case where the discharge water is fresh, i.e., with zero salinity. In this simplification, since no volume flux is taken into account in the flux calculation, (5.24) cannot be applied to simulate the water transport due to groundwater at the bottom. A volume flux at the bottom must be included for a realistic application.

5.5.2. A Complete Form of the Groundwater Input

The complete groundwater input module built into FVCOM allows users to add both volume and salt fluxes at the bottom. This module includes the volume change in the continuity equation due to the groundwater input and the salt change in the salinity equation. Groundwater flux (units: m^3/s) and salinity are required to be specified at nodes where the groundwater sources are.

The governing equations of FVCOM now include the groundwater input from the bottom. The code has been tested using benchmark problems for idealized cases.

Chapter 6: Open Boundary Treatments

One of the difficulties in applying an ocean model to a coastal region is how to specify a proper open boundary condition that allows the momentum or mass to be radiated out of or flow into the computational domain. Based on a criterion of a minimum reflection rate at open boundaries, Chapman (1985) made a comprehensive evaluation of numerical methods for the open boundary used in finite-difference models. Since the last FVCOM workshop held in June 2005, two major modifications were made to improve the open boundary treatment in FVCOM. First, we have built an open boundary radiation module with the inclusion of popular open boundary condition treatment methods. Second, we have extended our finite-volume open boundary treatment method to include the subtidal forcing and cross-boundary flux on the open boundary. A brief description of these open boundary treatment methods is given below.

6.1. Original Setup of the Open Boundary Treatment

In the original version of FVCOM, only two types of open boundaries were considered: one is for the case with specified tidal elevation at the open boundary and the other is for the case in which the free surface elevation is unknown at the open boundary.

In the first case, the velocity at the centroid of the boundary momentum control volume (MCE) is calculated through the linear momentum equations without inclusion of vertical and horizontal diffusion terms. The tracer values (e.g., water temperature and salinity) at the nodes of individual boundary tracer control elements (TCE) are calculated through three steps (Fig. 6.1). First, we calculate the flux out of or into individual MCE's at the boundary through the continuity equation given as

Fig.6.1: Illustration of the flux calculation at the open boundary for given tidal elevation at node points of boundary triangles.

$$F_C = \frac{\partial \zeta}{\partial t}\Omega^u - (F_A + F_B).\qquad(6.1)$$

Second, after the flux is determined, we calculate the vertically-averaged water temperature ($\bar{T}$) based on a net vertically-averaged flux through the boundary of a tracer control element (TCE).

Third, we use a gravity-wave radiation boundary condition to calculate $T' = T - \overline{T}$ at the node of the individual TCE in each layer in the vertical. The same approach can be used to calculate the boundary value for salinity (S) and other tracer values.

This method can guarantee to radiate the fastest surface gravity wave energy out of the computational domain and to ensure the mass conservation throughout the water column. In a case with complex bathymetry at the boundary, a sponge layer is usually specified around this area with a damping zone weighted from the open boundary into the interior with a specified influence radius to ensure that the radiation condition will also suppress the noise perturbation wave energy reflected back into the computational domain. This method works for the application of FVCOM to the Gulf of Maine/Georges Bank region and has shown to be stable for long-time integrations (seasonal to years).

It should be noted here that in order to reduce the error due to interpolation, we recommend that the triangles at the open boundary are constructed in the shape shown in Fig. 6.2. In this way, no interpolation from surrounding points is required when the gravity radiation boundary condition is applied. Also, in the FVCOM code, the perturbation values of T and S at the open boundary are calculated using the implicit gravity wave radiation condition described in Chen (1992). However, FVCOM is written such that users can apply other methods as desired.

Fig. 6.2: Illustration of the recommended shapes of triangular grids at the open boundary.

In the second case, the open boundary condition treatment is very similar to those used in the first case, except adding a gravity radiation condition for the free surface before the rest of the calculations are carried out. In the FVCOM code, the implicit gravity radiation condition described in Chen (1992) is used. This is consistent with the mass flux calculation used to determine the total flux through individual boundary MCE's. Also, any other radiation conditions that work for finite-difference models can be applied to FVCOM to replace the current gravity radiation condition.

6.2 Popular Radiation Open Boundary Conditions

Some model validation experiments have shown that the gravity wave radiation condition coded in the original version of FVCOM can cause the decrease of the sea level in the entire computational domain due to the wave reflection. This evidence is consistent with the analysis made by Chapman (1985), because the gravity wave radiation condition is not guaranteed to be non-reflecting in shallow water, particularly for river plume simulation. There is no unique radiation condition that works for all cases. To provide users more choices, we have implemented five popular open boundary conditions (OBCs) into FVCOM. In each OBC, the surface elevation at the open boundary is calculated by a radiation condition and the velocity in the open cell is computed using the linear or nonlinear momentum equations. The first method is the same as that used in the original OBC setup of FVCOM. In the second method, the vertically integrated flux at the open boundary is first computed according to the mass conservation law in the continuity equation and the vertically averaged velocity in the external mode is then calculated using the fully nonlinear momentum equations. The perturbation velocity, which is defined as the difference between the 3-D and 2-D currents, is determined using the linear momentum equations. In order to calculate the flux at the open boundary, the ghost cells are added at the open boundary in which the velocity is specified as the same value and direction in the open boundary cell. In fact, the perturbation velocity is solved in the no-gradient condition. The list of these five radiation conditions are given in Table 6.1.

Table 6.1: FVCOM's OBCs for the Surface Elevation

Type 1	**Active (ASL)** Sea level is specified at the OB. For example, tidal amplitude and phase (original FVCOM setup).
Type 2	**Clamped (ASL-CLP) (Beardsley and Haidvogel (1981)** $\zeta=0$ at OBC.
Type 3	**Implicit Gravity Wave Radiation (GWI) (Chapman, 1985)** $\zeta_t + C_o \zeta_n = 0$; $C_o = \sqrt{gH}$, n is the normal direction to the OB.
Type 4	**Partial Clamped Gravity Wave Radiation (BKI) (Blumberg and Kantha, 1985)** $\zeta_t + C_o \zeta_n = -\zeta / T_f$; $C_o = \sqrt{gH}$ T_f : user specified frictional timescale.
	Explicit Orlanski Radiation (ORE) (Orlanski, 1976; Chapman, 1985)

Type 5	$\zeta_t + C_o\zeta_n = 0; \; C_o = \begin{cases} \dfrac{\Delta n}{\Delta t} & if & -\dfrac{\zeta_t}{\zeta_n} \geq \dfrac{\Delta n}{\Delta t} \\[2mm] -\dfrac{\zeta_t}{\zeta_n} & if & -\dfrac{\zeta_t}{\zeta_n} < \dfrac{\Delta n}{\Delta t} \\[2mm] 0 & if & -\dfrac{\zeta_t}{\zeta_n} \leq 0 \end{cases}$

The four new OBC codes have been checked by running the following validation experiment for a freshwater discharge case over an idealized shelf (Fig. 6.3). The computational domain is constructed with a semi-enclosed rectangular basin with a length of 800 km, a width of 400 km. The water depth is 10 m at the coast and increases to 100 m over a cross-shelf distance of 100 km. The model is configured with the unstructured triangular grid with an open boundary located in the
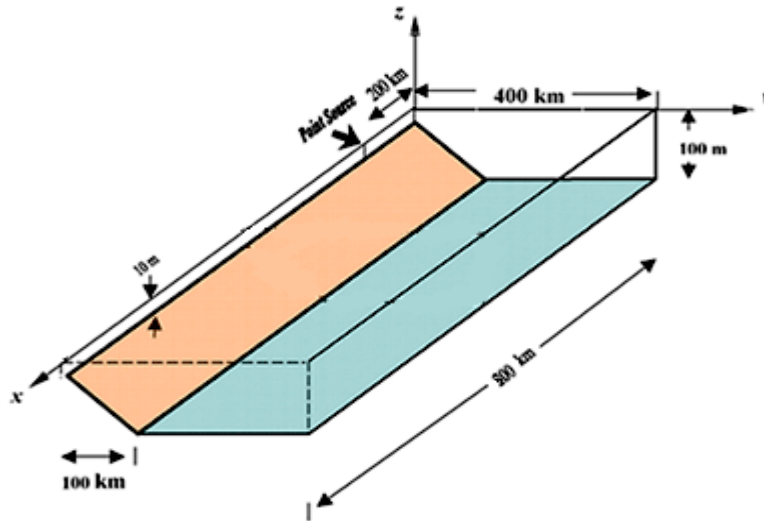
Fig. 6.3: The idealized, straight coastline continental shelf domain used for the OBC case tests.
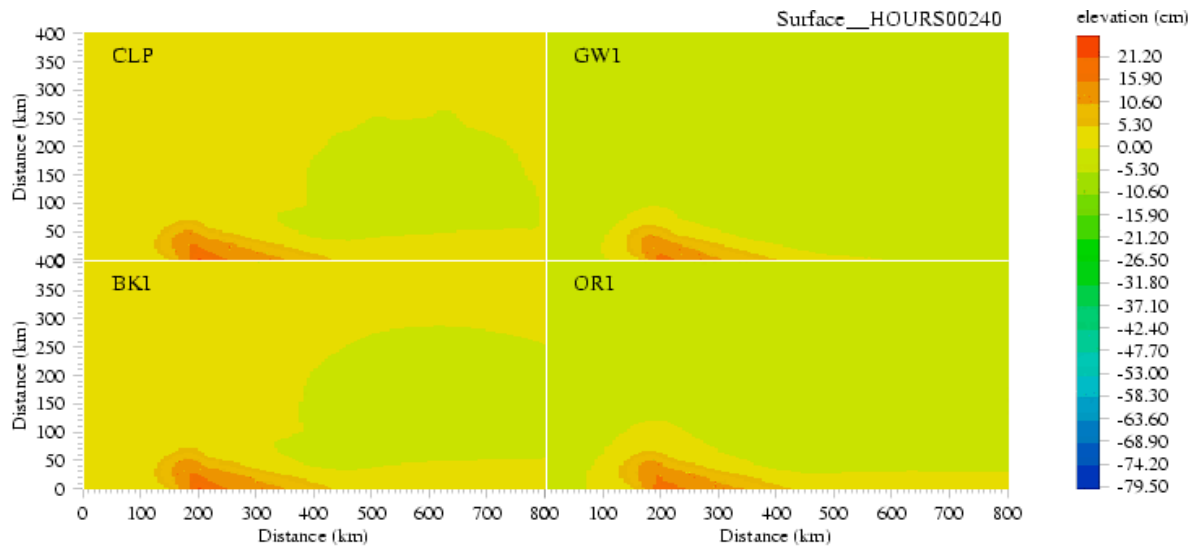
Fig. 6.4: Comparison of spatial distributions of the water level at the end of the 10th model day: (a) CLP; (b) GWI; (c) BKI; and (d) ORI.

downstream region 600 km away from the freshwater source located at x = 200 km. The
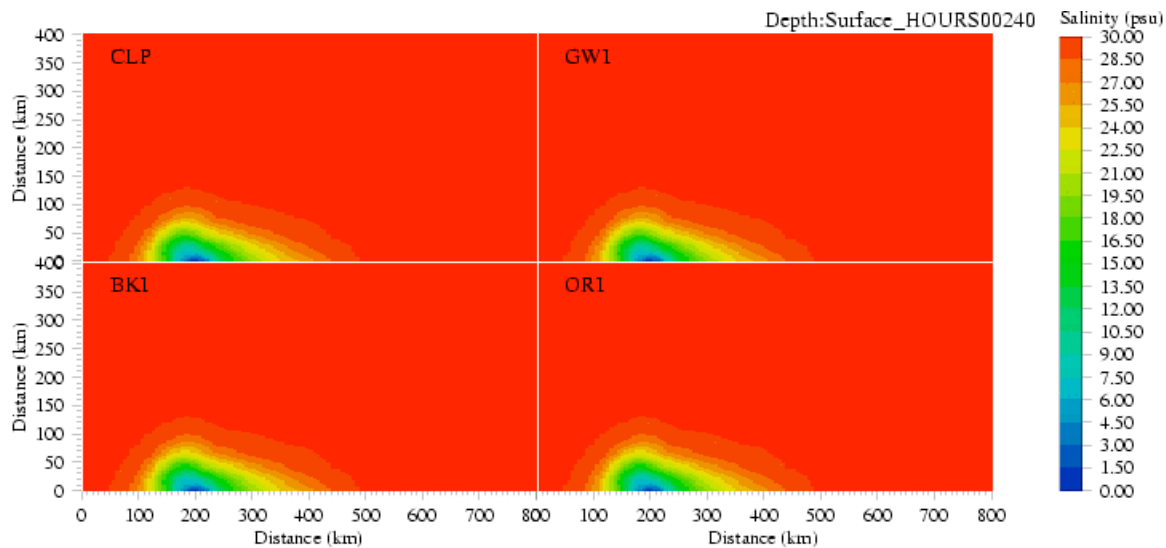
Fig. 6.5: Comparison of spatial distributions of the surface salinity at the end of the 10th model day: (a) CLP; (b) GWI; (c) BKI; and (d) ORI.

horizontal resolution is ~20 km and 10 sigma levels are used in the vertical with a vertical resolution ranging from 1 m at the coast to 30 m off the shelf. Freshwater discharge rate is 1000 m³/s. For initial conditions, a constant uniform value of salinity (30 PSU) was specified.

Model runs were made using the Type 2, 3, 4, and 5 OBCs. The surface water level, salinity and current fields at the end of the 10th model day are shown in Figs. 6.4 – 6.6. The results are summarized as follows:

CLP: No significant reflection was found in the first 10 days. When the plume reaches the OB, the reflection generates a flow along the OB. This artificial flow seems to have minor influence on the sea level and salinity in the interior in a time scale of 50 days. This is consistent with Chapman's (1985) finding. This minor influence should have little significant effect for a short-time model run.

GWI: The distributions of velocity and salinity look very similar to those predicted by the CLP condition. However, the sea level in the entire domain drops with time. On the 50th day, the sea level drop exceeds 0.5 m. GWI is not recommended for the river discharge case.

BKI: The sea level variation is the same as that predicted by CLP, but the velocity field is better. The distribution of the salinity is very similar to CLP and GWI.

ORE: The distributions of velocity and salinity are very similar to those predicted by BKI. However, as the same as GWI, the sea level drops with time. The rate of the sea level drop is smaller than that found in GWI.

In this simple test case, no significant differences were found for the choice of linear and nonlinear methods to calculate the velocity in open boundary cells. The nonlinear effects become critically important in an estuarine application when an open boundary is specified in the area with the intertidal zone.
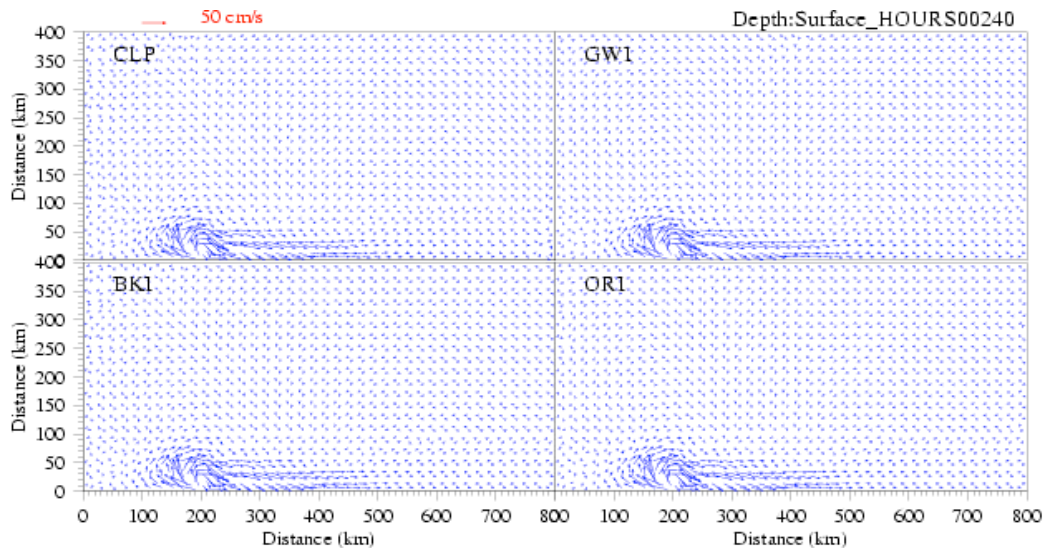
Fig. 6.6: Comparison of spatial distributions of the surface velocity at the end of the 10th day:
(a) CLP; (b) GWI; (c) BKI; and (d) ORI.

6.3 A New Finite-Volume Open Boundary Condition Module

By including various types of the radiation condition, FVCOM can be run for two types of numerical experiments: one is for the case with specified tidal harmonic elevations (amplitudes and phases of tidal constituents) at open boundary nodes, and the other is for the case with the free surface elevation calculated using various radiation condition at the open boundary nodes . The former is an active open boundary condition which is commonly used in coastal simulation experiments in a tidal environment. Because the surface elevation is specified at the open boundary node, however, this method does not allow any implementation of the time dependent volume flux and the setup of the subtidal elevation generated by other physical processes such as wind stress, heat and moisture fluxes, and density-adjusted geostrophic flow at the open

boundary. It might be ok if one is interested only in the circulation in the interior, but it definitely restricts its application to the regional ocean, particularly for the continental slope where either the western boundary current or shelfbreak front is located. An effort has been made to develop a new finite-volume open boundary condition for FVCOM that allows us to include both tidal and subtidal forcing at the open boundary. A brief description of the algorithm used to derive this new open boundary condition is given below.

Assuming all variables in the 2-D external mode equations can be decomposed into tidal and subtidal components as follows:

$$\zeta = \zeta_T + \zeta', U = U_T + U', V = V_T + V'$$
$$D = H + \varsigma_T + \varsigma' = D_T + \varsigma'$$

(6.2)

where $U = \int_{-1}^{0} u \, d\sigma$, $V = \int_{-1}^{0} v \, d\sigma$; and the subscript T denotes the tidal component and superscript ' denotes the subtidal component. Substituting (6.2) into the continuity equation yields

$$\frac{\partial \zeta_T}{\partial t} + \frac{\partial \zeta'}{\partial t} + \frac{\partial U_T D_T}{\partial x} + \frac{\partial U_T \varsigma'}{\partial x} + \frac{\partial U' D}{\partial x} + \frac{\partial V_T D_T}{\partial y} + \frac{\partial V_T \varsigma'}{\partial y} + \frac{\partial V' D}{\partial y} = 0$$

(6.3)

Because the tidal motion satisfies the continuity equation itself, eq. (6.3) can be simplified to

$$\frac{\partial \varsigma'}{\partial t} + \frac{\partial U_T \varsigma'}{\partial x} + \frac{\partial U' D}{\partial x} + \frac{\partial V_T \varsigma'}{\partial y} + \frac{\partial V' D}{\partial y} = 0$$

(6.4)

Integrating Eq. (6.4) over a TCE at the open boundary (see Fig. 6.7) produces

$$\iint_{\Omega_c} \frac{\partial \zeta'}{\partial t} dx dy = -[\oint_s (\vec{U}_T)_n \zeta' ds + \oint_s (\vec{U}')_n D ds] \quad (6.5)$$

where $(\vec{U}_T)_n$ and $(\vec{U}')_n$ are the tidal and subtidal vertically integrated velocity components normal to the boundary of the triangle, respectively; Ω_c is the area of a TCE; and s is the closed trajectory comprised of the boundary of a TCE.

Eq. (6.5) indicates that the subtidal surface elevation at the open boundary node is determined by the interaction between the tidal current and subtidal
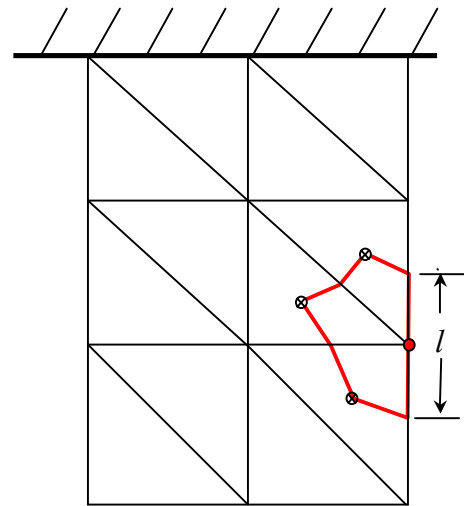
Fig. 6.7: Illustration of a TCE on the open boundary

elevation and the net flux driven by the vertically integrated subtidal current. The first term can be easily determined based on the known tidal currents, while the second term is only determinable when the subtidal current at the open boundary edge is known. Once the two terms on the right hand side of eq. (6.5) are computed, $(\zeta')^{N+1}$ can be easily derived using the same second-order four-stage Runge-Kutta time-stepping scheme as that configured in the external mode. Total surface elevation at the $N+1$ time step at a boundary node is then equal to

$$\zeta^{N+1} = \zeta_T^{N+1} + \zeta'^{N+1} \tag{6.6}$$

where ζ_T^{N+1} is the tidal elevation that is usually specified as the open boundary forcing and ζ'^{N+1} is the subtidal surface elevation that is determined by the mass conservative procedure.

The discrete approach to calculate the two terms on the right hand side of eq. (6.5) is described here. The first term at the time step N can be rewritten as

$$\oint_s (\vec{U}_T^N)_n \zeta' ds = \int_{s'} (\vec{U}_T^N)_n \zeta' ds + \int_l (\vec{U}_T^N)_n \zeta' ds \tag{6.7}$$

where s' is the trajectory of the TCE with a direct connection to the centroid of triangles of the TCE and l is the sideline length of the TCE on the open boundary. The first term on the right hand side of eq. (6.7) can be determined numerically by

$$\int_{s'} (\vec{U}_T^N)_n \zeta' ds = \sum_{m=1}^{NT} [(\Delta X_{2m-1}(V_T^N)_m - \Delta Y_{2m-1}(U_T^N)_m) \cdot (\zeta'^N)_m + (\Delta X_{2m}(V_T^N)_m - \Delta Y_{2m}(U_T^N)_m) \cdot (\zeta'^N)_m] \tag{6.8}$$

where NT is the total number of the sidelines connected to centroids of triangles in the open boundary TCE. The second term on the right hand side of eq. (6.7) can be determined numerically by

$$\oint_l (\vec{U}_T^N)_n \zeta'^N dl = (\vec{U}_T^N)_n \cdot \zeta'^N \cdot |\vec{l}| \tag{6.9}$$

where $|\vec{l}|$ is the length of l. $(\vec{U}_T^N)_n$ on l can be calculated inversely from the continuity equation of the tidal motion as

$$(\vec{U}_T^N)_n \cdot |\vec{l}| = -(\iint \frac{\partial \varsigma_T}{\partial t} dx\, dy + TF) / D_T^N \tag{6.10}$$

and

$$TF = \sum_{m=1}^{NT}[(\Delta X_{2m-1}(V_T^N)_m - \Delta Y_{2m-1}(U_T^N)_m)\cdot(D_T^N)_m$$
$$+ (\Delta X_{2m}(V_T^N)_m - \Delta Y_{2m}(U_T^N)_m)\cdot(D_T^N)_m] \qquad (6.11)$$

The second term on the right hand side of eq. (6.5) can be rewritten in a discrete form as

$$\oint_s (\vec{U}'^N)_n Dds = \oint_{s'} (\vec{U}'^N)_n Dds + \oint_l (\vec{U}'^N)_n D^N \, dl \qquad (6.12)$$

where

$$\oint_{s'} (\vec{U}'^N)_n Dds = \sum_{m=1}^{NT}[(\Delta X_{2m-1}(V'^N)_m - \Delta Y_{2m-1}(U'^N)_m)\cdot(D^N)_m$$
$$+(\Delta X_{2m}(V'^N)_m - \Delta Y_{2m}(U'^N)_m)\cdot(D^N)_m] \qquad (6.13)$$

and

$$\oint_l (\vec{U}'^N)_n D^N \, dl = (\vec{U}'^N)_n D^N\cdot|\vec{l}\,|. \qquad (6.14)$$

$(\vec{U}'^N)_n$ on l can be determined by two methods: one is to use a radiation boundary condition to calculate $(\vec{U}'^N)_n$ on the boundary and the other is to specify the subtidal flux on the boundary. A default setup for these two methods is described below.

I) BKI radiation condition

The Blumberg and Kantha implicit gravity wave radiation condition is chosen as the default setup for the first method. Let $\hat{U}_B$ be the value $(\vec{U}'^N)_n$ on the boundary, then

$$\hat{U}_B^{N+1} = [(1-\frac{\Delta t}{T_f})\hat{U}_B^N + c\frac{\Delta t}{\Delta n}\hat{U}_{B-1}^{N+1}]/(1+c\frac{\Delta t}{\Delta n}) \qquad (6.15)$$

where B is a boundary point and B-1 is the interior point next to the boundary point, and Δn is the distance between the interior and boundary point. This is a passive open boundary condition.

II) Flux condition

In many coastal applications, the subtidal flow field at the open boundary can be determined using either observational data or outputs from the regional or basin or global ocean models. For example, to run the East China Sea regional model, one must specify the volume flux of the Kuroshio on the slope at the open boundary. In this case, $(\vec{U}'^N)_n$ on l can be determined using the value specified by users. The volume flux produced by the subtidal current on the boundary can be specified by the same procedure used to add the river discharge on the boundary edge. For

detail, please refer to Chapter 5. We call this method an active open boundary condition, because it is controlled by users.

In the case without tides, this boundary treatment procedure is like a traditional radiation condition and the model can be run directly with this condition. In the case with both tidal and subtidal forcing, the model needs to run for the tidal forcing only first. By running the model for the only tidal forcing, we can determine the tidal currents at the centroid and at nodes of open boundary cells. These data are automatically output into a file as the input file for the model run with combined tidal and other forcing.

When a radiation boundary condition is specified for the subtidal current at the open boundary, a sponge layer might be required to filter the high frequency numerical noise due to the wave reflection.

This method is validated by running the model with this condition for idealized cases and applications to the real ocean. The first experiment was made to re-run the river discharge plume case shown in Fig. 6.3. The results are identical to that used in BKI radiation condition. For example, three idealized case results are shown below.

a) Channel Flow

This test case was used by investigators at the USGS Woods Hole Science Center to test their sediment module "SED_TEST1" in ROMS. Detailed description of the configuration can be found at http://woodshole.er.usgs.gov/project-pages/sediment-transport/Test_Case_1.htm. A brief description is repeated here.
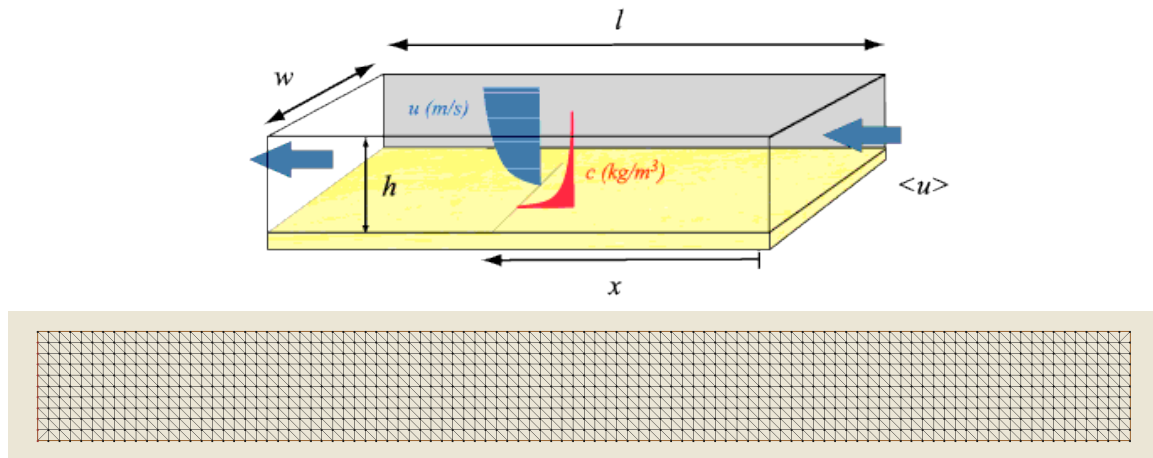
Fig. 6.8: (a) Schematic of channel flow test case. (b) Model grid used.

The model domain is a long, narrow rectangular box with length l = 10,000 m, width w = 1000 m, and uniform depth h = 10 m (Fig. 6.8a. This figure was directly downloaded from the USGS website). Fresh water with T = 20°C, S = 0 PSU flows in from the eastern open boundary (x = 0) and flows out from the western open boundary. There is no rotation, wind forcing, and heating/cooling involved. Inflow is maintained as a steady flow with volume transport of 10 m^3/s per meter of of width and uniformly distributed in the vertical. Northern and southern sides of the box are solid boundaries. Bottom roughness z_0 = 0.005 m.

The computational domain is configured with the triangular grid in the horizontal and sigma layers in the vertical. Horizontal resolution is 100 m and vertical resolution is 1 m (with 10 sigma layers). The fluid is initially at rest, and the model spins up with a ramping period of 4 hrs.
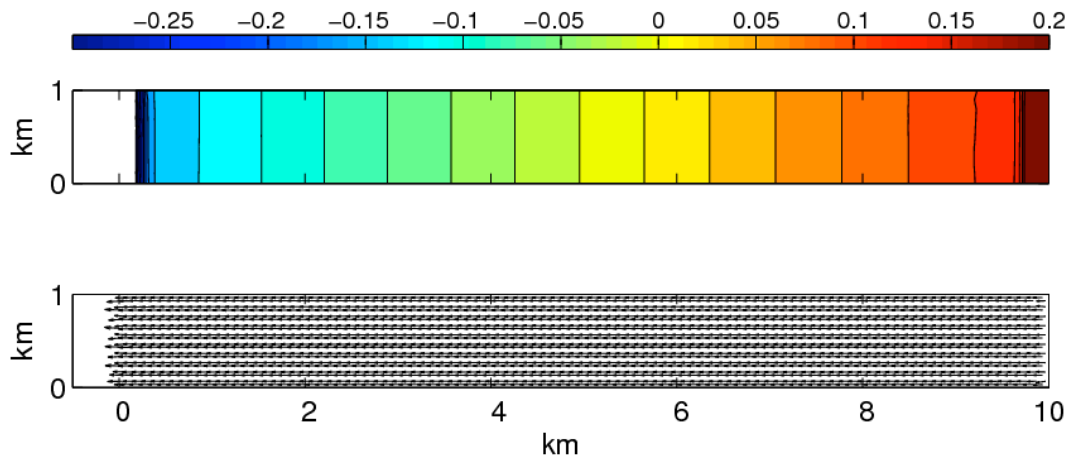
Fig. 6.9: (a) Sea surface elevation (m) and (b) surface layer velocity vectors at the 200th model hour. A sponge layer is added at the outflow open boundary.

The model runs stably with inflow and outflow conditions. Fig. 6.9 shows the sea surface elevation and velocity vectors in the surface layer at the 200th model hour. A sponge layer, with a friction coefficient increased from 0 to 0.05 at the open boundary over a distance of r = 400 m, is used to absorb disturbances and suppress computational noise at the outflow side. The sponge layer only affects the numerical solution near the boundary with no influence on the interior solution.

b) Flow in a Slope Bottom Channel.

This is a case used by investigators at the USGS Woods Hole Science Center to test their estuarine module "ESTUARY_TEST" in ROMS. Detailed information about the experiment

design   can   be   viewed   at   http://woodshole.er.usgs.gov/project-pages/sediment-transport/Test_Case_2.htm. A brief description is given below.

The model domain is a long, narrow rectangular channel with length (east-west) l = 100 km, width (north-south) w = 10 m, and depth changing linearly from h = 10 m at the western end (x = 0 km) to 5 m at the eastern end (x = 100 km). A constant inflow transport of 5 m^3/s per meter of width is specified at the eastern end and the same amount of outflow transport is specified at the western end. The M_2 tidal forcing with amplitude of 10 cm is specified at the western end. The experiment was made for the barotropic case in which salinity and temperature were set to the same constant values at the inflow and outflow boundaries. No rotation, wind forcing and heating/cooling were included. Both lateral sides are treated as inviscid solid walls. The model was run for the fully nonlinear case with bottom roughness z_0 = 0.005 m.
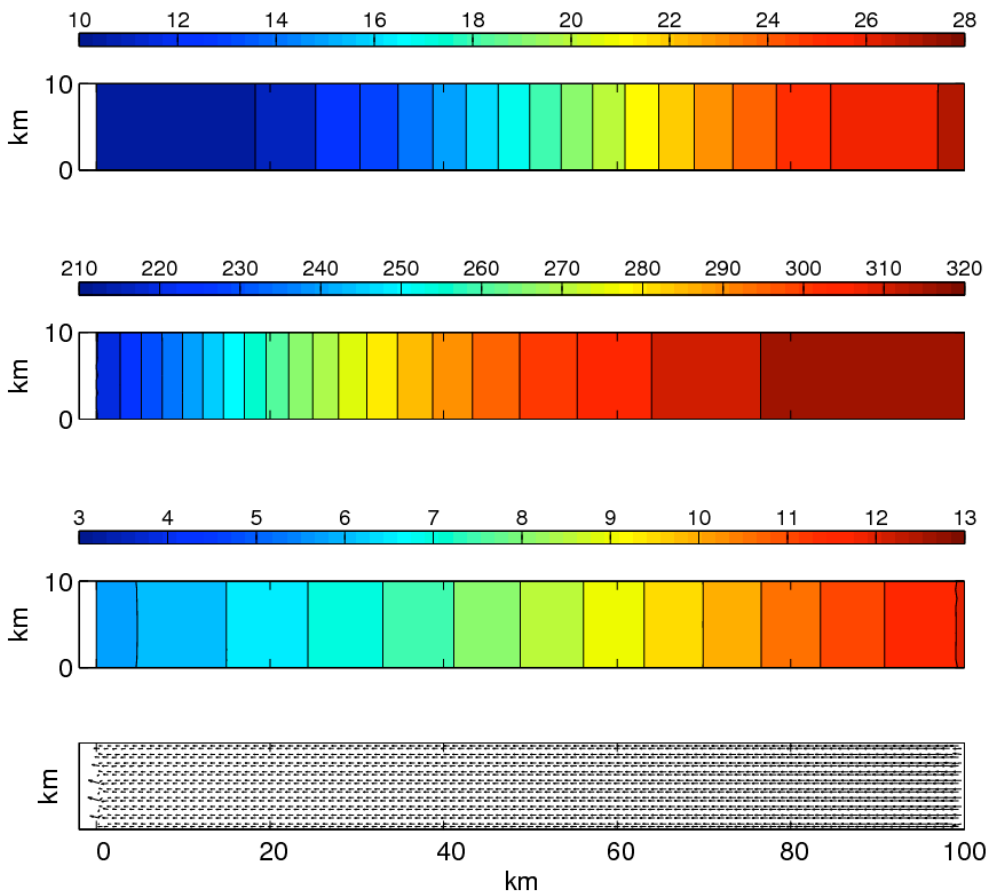
Fig. 6.10: (a) M_2 amplitude (in cm), (b) M_2 phase (°), (c) ζ (in cm), and (d) surface layer residual velocity vector for the estuary test case at day 30.

Simulation results are shown in Fig. 6.10. We can see that the M_2 elevation amplitude and phase are well simulated (Fig. 6.10a-b), which resemble those shown in the case with only tidal forcing (figures not shown here). The residual water elevation (ζ) (Fig. 6.10c) and residual surface velocity vector (Fig. 6.10d) are also similar to those shown in the case with the only inflow and outflow transports (figures are not shown here).

c) Flow over an Idealized Continental Shelf

Consider an idealized continental shelf constructed by a zonal channel that is 600 km long and 200 km wide (Fig. 6.11). The bathymetry is constant in the along-shelf direction and has a hyperbolic tangent profile in the cross-
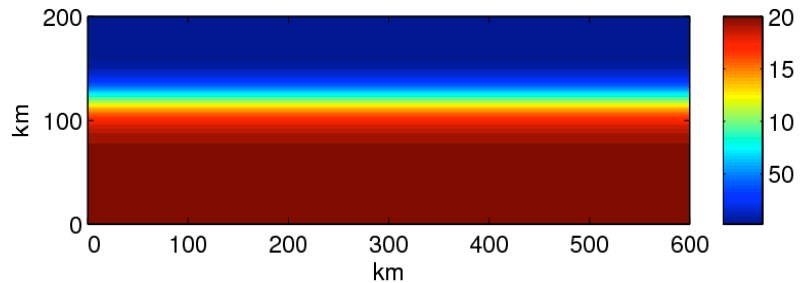
Fig. 6.11: Distribution of bottom depth (in m) in the continental shelf case.

shelf direction. The coastline is located on the southern (shallow) side which is treated as a solid boundary and all other three sides are open boundaries. Assuming that the tide propagates shoreward from the open ocean, the tidal forcing is only specified at the offshore open boundary. An along-shelf current with a volume transport of 7.35 Sv is prescribed at the shelf break. The experiment was carried out for a barotropic case in which the water temperature and salinity remain constant in time.

The computational domain is configured with the triangular grid with horizontal resolution of 10 km. A total of 10 sigma layers are used in the vertical. The model was spin up over a ramping period of 24 hrs for the case with only tidal forcing and 48 hrs for the case with both tidal and subtidal forcing.

Figs. 6.12 and 6.13 show the model-predicted amplitude and phase of the M_2 elevation and the sea surface elevation and current for the case with only tidal forcing and for the case with only inflow/outflow transport, respectively. Fig. 6.14 shows the model result for these variables for the case with both tidal forcing and inflow/outflow transport. The basic patterns are very similar to the model runs with separate forcing. The difference of M_2 elevation phase near the eastern boundary between Fig. 6.14 and Fig. 6.12 is probably due to the time series interpolation (used to force the model) or the nonlinear interaction between tidal waves and mean flow.
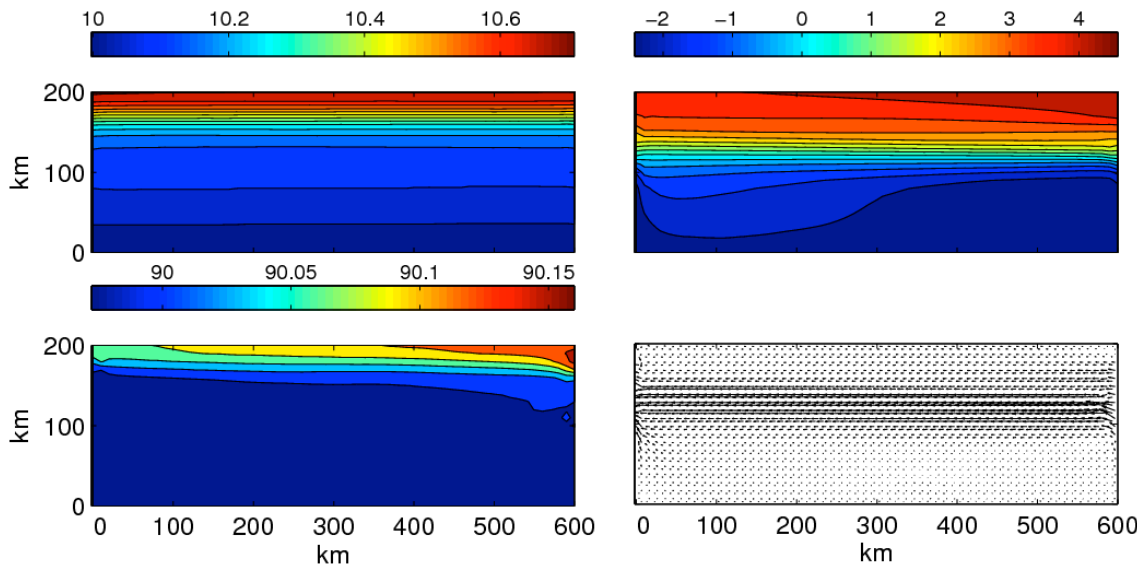
Fig. 6.14: Left panel: distributions of the M2 amplitude (in cm) (upper) and phase (°) (lower). Right panel: distributions of the surface level (in cm) (upper) and surface layer residual velocity vectors (cm/s) (lower) for the case forced by tidal and subtidal flow flux at the open boundary.

6.4 Nesting Boundaries

Conceptually, grid refinement techniques, such as nesting, conjoined grids and adaptive grids, could be employed to endow ocean models with variable resolution capabilities, and to permit these models to better resolve multi-scale oceanic processes. One-way or two-way nesting is a common approach used in both atmosphere and ocean models, but this approach has limitations, especially at the seam where the two different-size grids are connected and where one cannot ensure continuity in phase and group speed of propagating waves. For example, in a free-surface shallow-water ocean model, under a long-wave approximation, the surface gravity waves are non-dispersive with phase speed $\sqrt{gH}$. On a discrete grid, however, the model-simulated gravity wave phase speed is dispersive and depends on horizontal resolution (see Fig. 6.15 for an example derived from a simple finite-difference scheme). Since the grid sizes of the two domains differ at the nesting boundary, the model-computed phase and group speeds experience a jump. Special treatment is then required to disperse inconsistent energy exchange between the two grids, and to ensure mass and energy conservation at the nesting boundary.
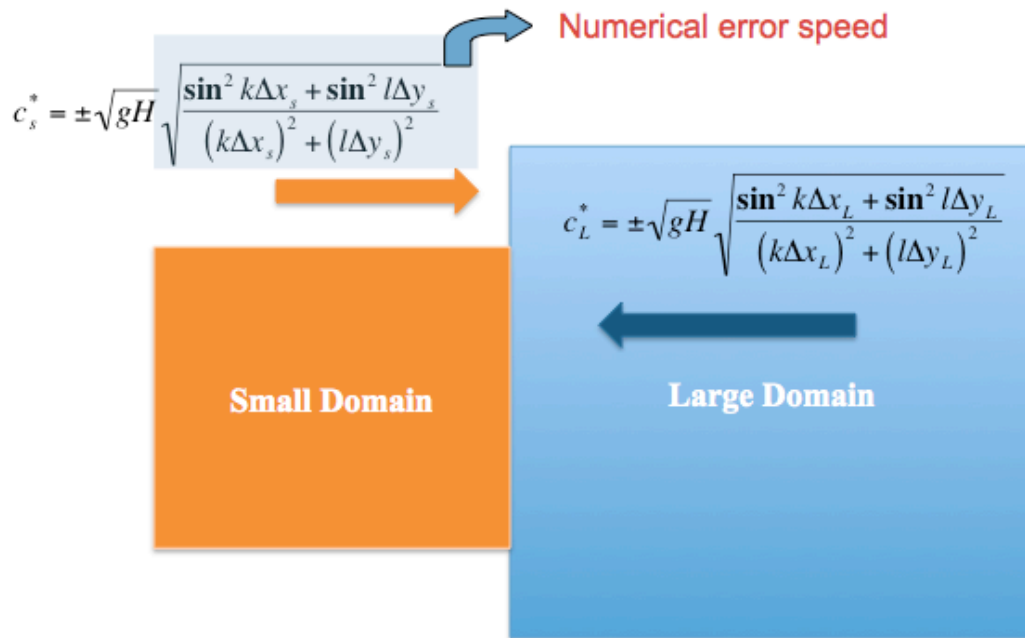
Fig. 6.15: Illustration of nesting two domains in a structured-grid model. Because $\Delta x_s \neq \Delta x_L$; $\Delta y_s \neq \Delta y$; the surface gravity wave phase and group speeds at the nesting boundary in the two domains are not equal.

Analogous baroclinic dynamics face the same issue. This treatment usually works for a short-term simulation but has not been tested in the long time-scale simulations generally used in climate modeling, primarily because of the associated computational complexities and expense.

Unlike the structured-grid nesting discussed in Fig. 6.15, unstructured-grid nesting is much simpler. By linking two domains with common cells, this approach produces the same surface gravity wave characteristics at the boundary, which ensures volume and mass conservation between the two domains (Fig. 6.16). This approach also makes it practical to resolve multi-scale processes in the ocean. For example, energetic high-frequency internal waves are frequently generated over submarine banks, seamounts, and ridges. We can create a finer grid over those regions and nest it to the regional model. In this small sub-domain, we can turn on the non-hydrostatic dynamics in the sub-domain and then hydrostatic dynamics in the regional domain. Using the varying-size grids to specify the nested boundary with coarser grids in the region where the hydrostatic dynamics are dominant (see the green and yellow area in Fig. 6.16), we can run the multi-scale coupling in FVCOM in a computational efficient way.
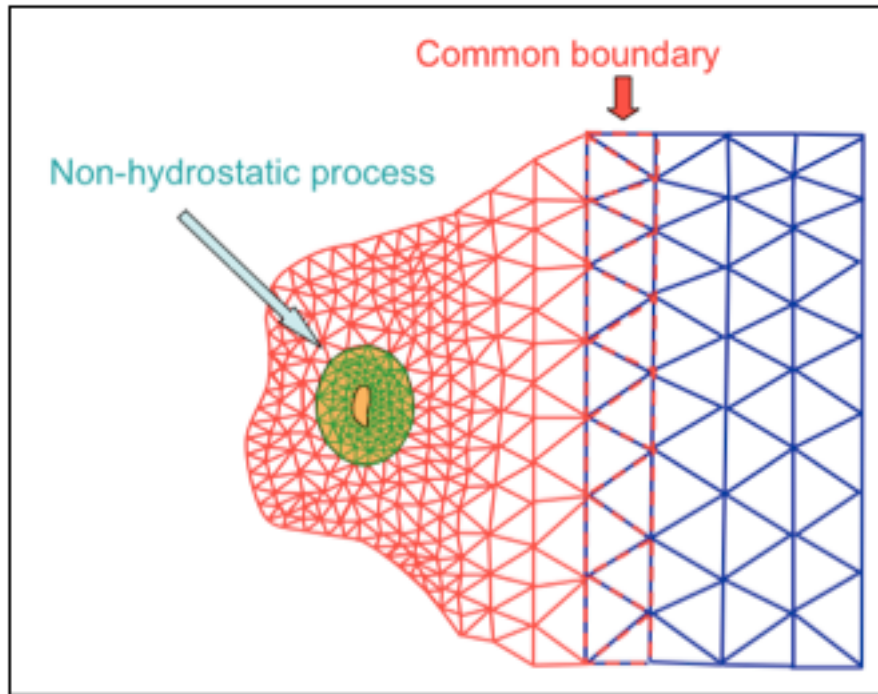
Fig. 6.16: Schematic of nesting in FVCOM. With common cells between nested domains, it can avoid the numerical energy accumulation and ensure the volume and mass conservation.

We have implemented three types of one-way nesting boundaries in FVCOM v3.1.6 or up. The first two are used for FVCOM multi-domain nesting and the third one is used to nest FVCOM with a structured-grid model.   These types of nesting are defined as "direct nesting", "indirect nesting", and "relaxation nesting".

For "direct nesting", the small domain FVCOM is driven directly with the nested boundary output from the large domain FVCOM. In this case, only the variables at boundary nodes and cells are required.

For "indirect nesting", the small domain FVCOM will retain its own tidal forcing at the nesting boundary and nest with the large domain FVCOM using the subtidal values of variables at boundary nodes and cells.   Because tides in the coastal region are controlled by local bathymetry and open boundary forcing, the large domain FVCOM-predicted tide might have a larger error than the small domain FVCOM. The indirect nesting can avoid the tidal simulation errors that might come into the small domain through the nesting boundary.

For "relaxation nesting", the nesting is set up in the relaxation zone. Since the variables from a structured-grid model are required to be interpolated to the nodes and cells in the nesting zone, the input values cannot ensure volume and mass conservation. Also, the model output from the large-domain structured-grid model might significantly differ from the small domain FVCOM model results in the boundary zone. If one directly replaces the FVCOM variable in the nesting boundary zone using the structured-grid model output, it can cause a discontinuous jump and hence lead to numerical instability. The relaxation zone is designed with two objectives: 1) to make sure that the gradients of variables (e.g., sea level, baroclinic pressure, temperature) from the large domain model can be included in the nesting, and 2) to make a transition to merge variables of the large-domain structured-grid model with variables of the small-domain
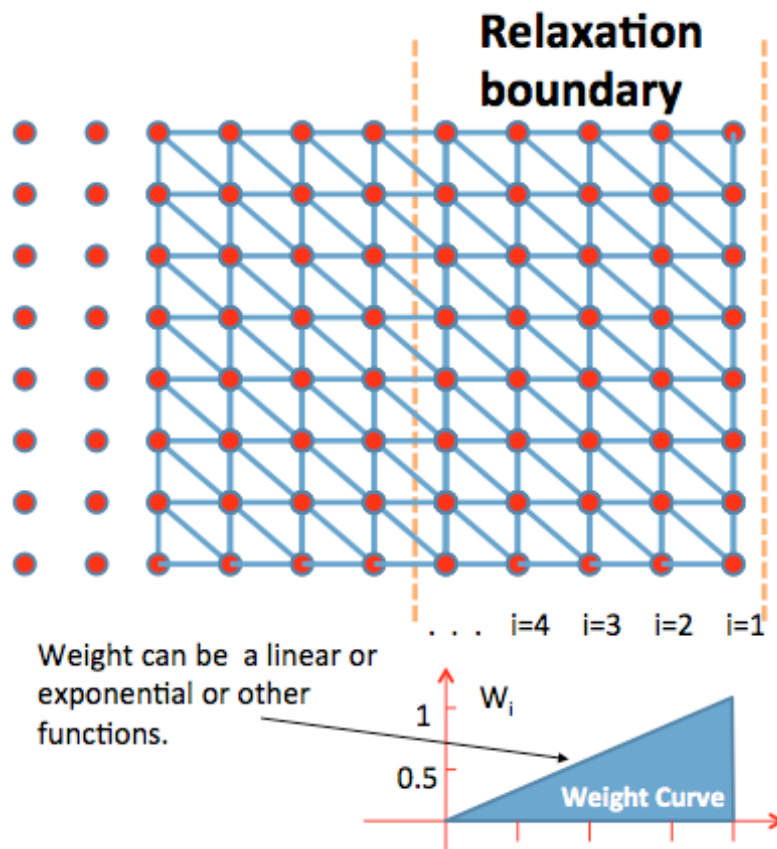
Fig. 6.17: Illustration of the relaxation boundary zone for "relaxation nesting".

unstructured-grid model. Including the gradients of variables in the nesting is critical for currents.

Defining U, V as the x- and y-components of velocity and ζ as the surface elevation, their values in the boundary zone are given as

$$U_{\text{boundary}(i)} = W_i U_{\text{nest}} + (1 - W_i) U_{\text{FVCOM}(i)} \tag{6.16}$$

$$V_{\text{boundary}(i)} = W_i V_{\text{nest}} + (1 - W_i) V_{\text{FVCOM}(i)} \tag{6.17}$$

$$\zeta_{\text{boundary}(i)} = W_i \zeta_{\text{nest}} + (1 - W_i) \zeta_{\text{FVCOM}(i)} \tag{6.18}$$

where W_i is the weight function and i counts from the outer edge boundary (see Fig. 6.17). The subscripts of "boundary", "nest" and "FVCOM" indicate the forcing values used to drive the model, nested values from the large-domain structured-grid model, and original values calculated from FVCOM. All other variables, which are required in the nesting boundary zone, use the same nesting functions listed in eqs. (6.16)-(6.18). The default setup of the weight function is linear. Users can change this function to be exponential or others. The relaxation-nesting module allows users to define the nesting boundary zone by either the distance or layers of triangular cells. The same approach is also used to define the weight function. For the coastal ocean model with tidal forcing, if users want to keep the tidal forcing of FVCOM and also include the subtidal components from a large-domain structured-grid model, then users can use the nesting functions (6.16) and (6.17) for subtidal components together with restoring of tidal velocity and sea level back in the nesting boundary zone. The relaxation-nesting module in FVCOM has considered this option.

Chapter 7: The Dike and Groyne Module

7.1 Introduction

It is a challenge for a terrain-following coordinate ocean model to simulate the flow field in an estuarine or coastal system with dikes and groynes. In the most cases, the structures are usually submerged during high tide but out of the water during low tide. If a vertical wall is placed within the computational domain, then the terrain-following coordinate transformation can fail. Adding a slope on the surface of a dike or groyne could make the topographic coordinate transformation work, but it changes the fluid dynamics. Instead of solid blocking (no flux towards the wall) in the lower column with the dike or groyne and free exchange in the upper column above the structure, the model makes the water flow along the submerged part of the structure under the dynamics of the sloping bottom boundary layer. As a result, this slope treatment could overestimate vertical and lateral mixing and thus produce unrealistic circulation around the structure.

Coastal inundation, which is defined as coastal flooding of normally dry land caused by heavy rains, high river discharge, tides, storm surge, tsunami processes, or some combination thereof, has received intense attention in model applications to coastal and estuarine problems. In many coastal regions, dams are built around the area where the height of land is lower or close to the mean sea level to protect the land from flooding. An inundation forecast system is aimed at making warning of coastal flooding on an event timescale in order to facilitate evacuation and other emergency measures to protect human life and property in the coastal zone. Such a system can also be used to estimate the statistics of coastal inundation in order to enable rational planning regarding sustainable land-use practices in the coastal zone. A model used for this application must produce accurate, real-time forecasts of water level at high spatial resolution in the coastal zone and must have the capability to resolve the overtopping process of dams. These dams are like a solid wall boundary when the water level is lower than it, but become submerged structures like dikes when flooding occurs. The wet/dry treatment technology in the original FVCOM is capable of resolving coastal flooding (Chen et al., 2006a,b), but cannot handle a vertical seawall in the computational domain.

We have developed an unstructured-grid dike and groyne treatment algorithm in a terrain-following coordinate system to calculate the velocity and tracer concentration in a coastal or estuarine system with emerged or submerged dikes and groynes. This algorithm has been coded into FVCOM with MPI parallelization, and validated for idealized cases with dike-groyne construction where analytical solutions or laboratory experiment results are available. The FVCOM with inclusion of dike and groyne treatment module has been applied to simulate the flow field off the Changjiang Estuary where a dike-groyne structure was constructed in the Deep Waterway channel in the inner shelf of the East China Sea and also to simulate coastal inundation in Scituate Harbor, Massachusetts. A full description of the FVCOM dike and groyne treatment algorithm and validation and application results were recently published by Ge et al. (2012). A brief description of this module (taken from Ge et al., 2012) is given below.

7.2 An Unstructured-Grid Dike-Groyne Algorithm

Consider a submerged dike or groyne case. For this case, the water column connected to the structure is characterized by two layers: an upper layer in which the water can flow freely across the structure, and a lower layer in which flow is blocked (with no flux into the wall). In general, the width of a dike or groyne is on the order of 2-5 m. For a numerical simulation with a horizontal resolution of > 20-100 m, these dikes or groynes can be treated as lines without width. Under this assumption, we can construct the triangular grid along dikes and groynes, with a single control volume above the structure and two separate control volumes beneath it (Fig. 7.1).

In the Cartesian coordinate system, the vertically integrated continuity equation can be written in the form of

$$\frac{\partial \zeta}{\partial t} = -\frac{1}{\Omega}[\oint_{l_\Omega}(\bar{u}D)\,dy - \oint_{l_\Omega}(\bar{v}D)\,dx] \tag{7.1}$$

where ζ is the free surface elevation, u and v are the x and y components of the horizontal velocity, D is the total water depth defined as $H+\zeta$, and H is the mean water depth. In FVCOM, an unstructured triangle is comprised of three nodes, a centroid, and three sides, on which u and v are placed at centroids and all scalars (i.e., ζ, H, D) are placed at nodes. u and v at centroids are calculated based on the net flux through the three sides of that

triangle (shaded regions in Fig. 7.1, hereafter referred to as the Momentum Control Element: MCE), while scalar variables at each node are determined by the net flux through the sections linked to centroids and the middle point of the sideline in the surrounding triangles (shaded regions in Fig. 7.1), hereafter referred to as the Tracer Control Element: TCE). Ω is the area of the TCE.

a) upper column   b) lower column
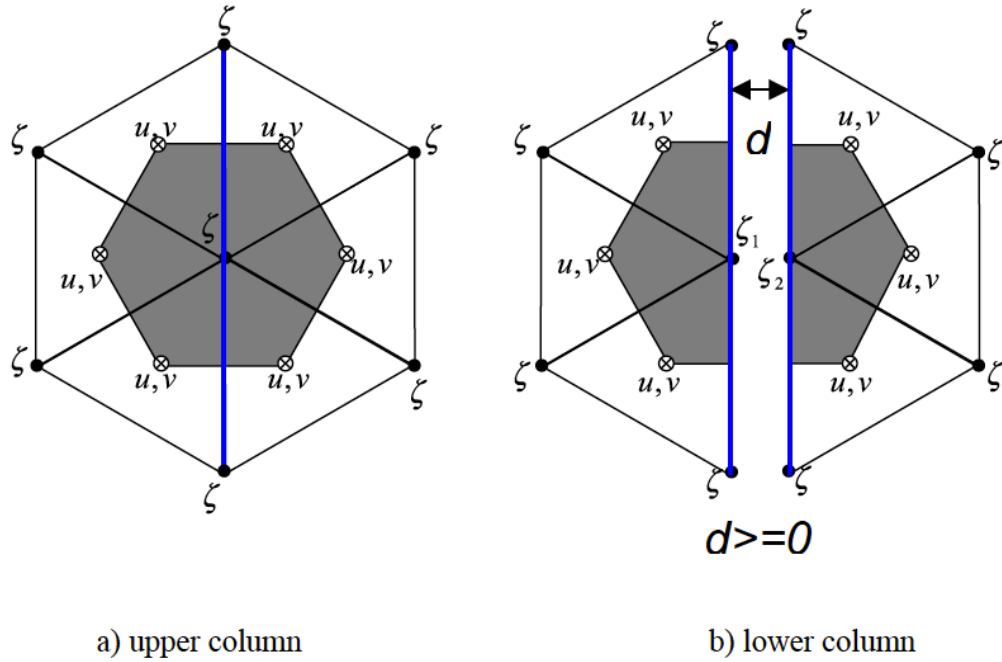
Figure 7.1: Sketch of the separation of the control element at dikes or groynes. The shaded regions indicate the tracer control elements (TCEs).

Defining h as the height of dike or groyne, we divide a TCE into two elements (Fig. 7.1), calculate the flux individually, and then combine them. Applying eq. (7.1) to each element, we have

$$\Omega_l \frac{\partial \zeta_l}{\partial t} = -[\int_{l_l} \bar{u}D\,dy - \int_{l_l} \bar{v}D\,dx] - [\int_{l_w} \bar{u}_w D\,dy - \int_{l_w} \bar{v}_w D\,dx] \tag{7.2}$$

$$\Omega_r \frac{\partial \zeta_r}{\partial t} = -[\int_{l_r} \bar{u}D\,dy - \int_{l_r} \bar{v}D\,dx] + [\int_{l_w} \bar{u}_w D\,dy - \int_{l_w} \bar{v}_w D\,dx] \tag{7.3}$$

where Ω_l and Ω_r are the areas of the two elements (hereafter referred to as left and right elements); l_w is the length of the element edge connected to the solid wall; l_l and l_r are

the lengths of left and right elements (minus l_w); ζ_l and ζ_r are the surface elevations calculated by the flux derived from the left and right elements; and u_w and v_w are the x- and y-components of the horizontal velocity at the edge of the element connected to the wall. u_w and v_w satisfy the boundary condition of no flux normal to the wall. The eqs. (7.2) and (7.3) are numerically solved using the modified fourth-order Runge-Kutta time-stepping scheme, the same as that used in FVCOM (Chen et al., 2003a; 2006d).

For the case in which the dikes and groynes remain under the sea surface, adding eqs. (7.2) and (7.3) yields

$$\Omega_l \frac{\partial \zeta_l}{\partial t} + \Omega_r \frac{\partial \zeta_r}{\partial t} = -[\int\limits_{l_r + l_r} \bar{u} D \, dy - \int\limits_{l_r + l_r} \bar{v} D \, dx] \tag{7.4}$$

According to volume conservation, we can determine ζ at the node on the wall with a solution given as

$$\zeta = \frac{\Omega_l \zeta_l + \Omega_r \zeta_r}{\Omega_l + \Omega_r} \tag{7.5}$$

Eq. (7.5) is derived for a submerged dike or groyne case. For the case in which dikes and groynes are initially above sea level, the surface elevation on either side of the wall is determined by ζ_l and ζ_r in eqs. (7.2) and (7.3). Under a condition of total water depth D on both sides being higher than the height of the wall, the surface elevation can be calculated by eq. (7.5). When the total water depth on one side is higher than the height of the wall but on the other side is not, then the volume of the water above the height of the wall will move to the other side as a lateral flux. For example, assuming that the water on the left side, but not on the right side, is higher than the height of the wall (Fig. 7.2), i.e.,

$$D_l = H + \zeta_l > h; \; D_r = H + \zeta_r < h \,,$$

then the new surface elevations on the respective sides should be equal to

$$\hat{\zeta}_r = \zeta_r + \Delta \zeta_l \frac{\Omega_l}{\Omega_r} \quad \text{and} \quad \hat{\zeta}_l = \zeta_l - \Delta \zeta_l \,. \tag{7.6}$$

If the adjusted total water depth $D_r = H + \hat{\zeta}_r > h$, then a revised adjustment is made until ζ_l equals ζ_r. This approach is also applied in the case where the mean depths on opposite sides of the wall are different.

In FVCOM, the horizontal velocity is calculated using the second-order upwind scheme derived by Kobayashi et al. (1999). This method was described in detail in Chen et al. (2003a). When the dikes and groynes remain above sea level, then they are treated as a solid lateral boundary, and velocity at the centroid of a triangle connected to the wall can be easily determined using the same boundary treatment as in FVCOM (Chen et al., 2006d). For the case in which the dikes and groynes are below sea level, the velocity in the upper free-exchange ($-H + h \le z \le 0$) and lower solid-blocking ($-H \le z < -H + h$) layers are calculated based on the MCEs shown in Fig. 7.1(*a* and *b*), respectively. No flux normal to the wall is applied to the MCE in the lower layer.
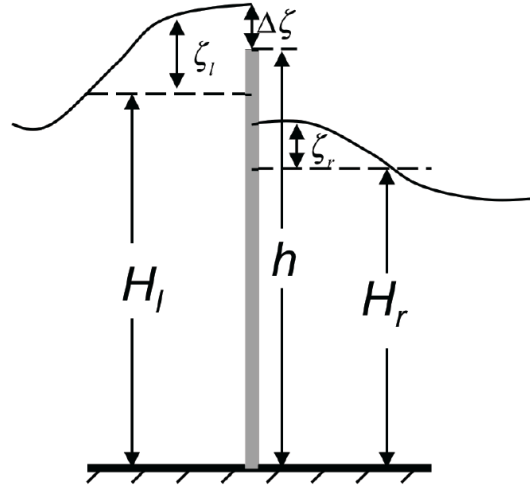
Figure 7.2: Illustration of the treatment of the water exchange across a dike or groyne when the water on either side is over the height of the structure.

The governing equations in FVCOM are solved using either a semi-implicit scheme or a mode-split scheme. In the semi-implicit scheme, the velocity can be solved using the approach described here. In the mode-split scheme, the total water flux over the wall equals $(D - h)\bar{v}_n$, where $\bar{v}_n$ is the component of vertically averaged velocity normal to the wall. This amount of transport must be considered in the 2-D mode to be consistent with the 3-D calculation.

The vertical velocity (ω) in the terrain-following vertical coordinate is calculated based on the same TCEs as those used for the surface elevation. In the lower solid-blocked layer, ω on the sides of the wall is determined by the left and right TCEs shown in Fig. 7.3b, while in the upper free-exchange layer, it is calculated using the combined TCE shown in Fig. 7.3a, i.e.,

$$\omega_{i,k+1} = \omega_{i,k} + \frac{\Delta\sigma_k}{\Delta t_i}(\zeta_i^{n+1} - \zeta_i^n) + \frac{\Delta\sigma_k}{\Omega_l + \Omega_r} \oint_{l_l+l_r} u_{N,k}^n Ddl \ , \tag{7.7}$$

and in the lower column, the velocity at the vertical level in the two TCEs are calculated separately, as

$$\begin{cases} \omega_{i,k+1}^l = \omega_{i,k}^l + \frac{\Delta\sigma_k}{\Delta t_i}(\zeta_i^{l,n+1} - \zeta_i^{l,n}) + \frac{\Delta\sigma_k}{\Omega_l} \oint_{l_l} u_{N,k}^n Ddl \\[4mm] \omega_{i,k+1}^r = \omega_{i,k}^r + \frac{\Delta\sigma_k}{\Delta t_i}(\zeta_i^{r,n+1} - \zeta_i^{r,n}) + \frac{\Delta\sigma_k}{\Omega_r} \oint_{l_r} u_{N,k}^n Ddl \end{cases} \tag{7.8}$$

where ω^l and ω^r are the vertical velocities at the separate left and right TCEs and k is the vertical level index.

a) upper column   b) lower column
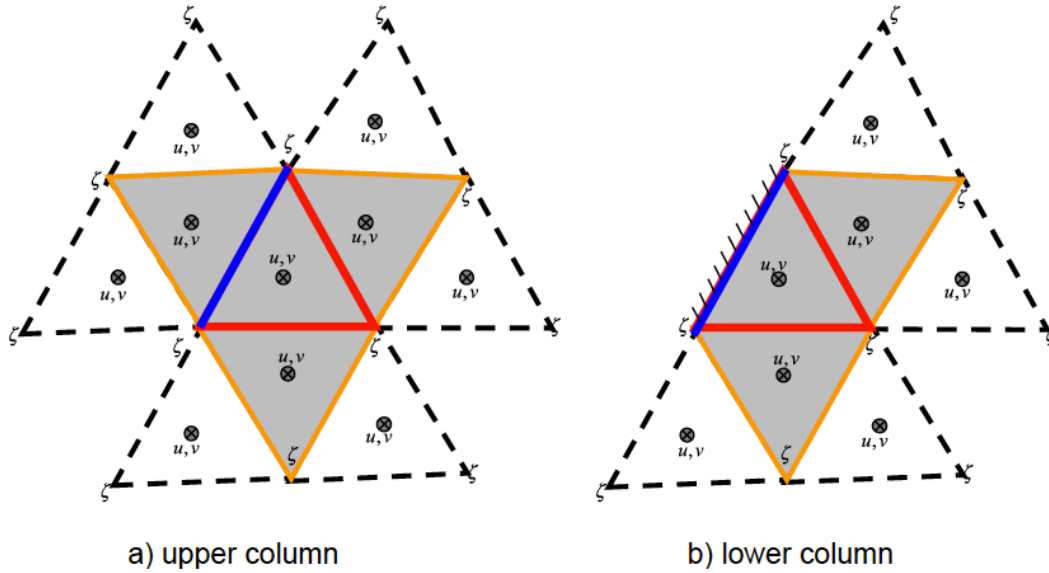
Figure 7.3: Illustration of momentum control elements (shaded regions) used to calculate the horizontal velocity in the upper (above the height of the structure) and lower (below the height of the structure) layers.

The calculation of scalar variables at nodes with triangles connected to dikes and groynes is similar to that used for the surface elevation and vertical velocity. A special treatment is made for the case in which the water is moved from one side (where the total water depth is greater than the height of the wall) to the other side (where the total water depth is less than the height of the wall). For example, in the case indicated in eq. (7.6), the $\Delta\zeta_l\Omega_l$ water is removed from the left TCE and added to the right TCE. If T_l is the

water temperature in the left TCE, then an adjustment will be made to extract $\Delta \zeta_l \Omega_l T_l$ from the left TCE and add it to the right TCE in the flux calculation of the temperature equation. The same approach is used for salinity and other scalar variables.

7.3. Examples of Validation Experiments

Case I: An idealized overtopping problem. Consider an overtopping problem in a rectangular channel with length = 5 km ($2L$) and width = 1 km (D). A 10-m high (H) vertical seawall is placed at the shoreline at the mid-point ($x = 0$) (Fig. 7.4a). The ocean side ($x > 0$) features a flat bottom channel filled fully with water, while the landside ($x <$ 0) is characterized by a linear slope that is initially dry. The maximum height of the shore is 10 m, the same height as the seawall. The origin of the vertical coordinate ($z = 0$) is defined at the reference water level at the top of the seawall. The model was run with a constant discharge with rate Q, which is specified uniformly in the vertical at the open boundary (x = L).

Let $t = 0$ at the start of the model run, so that the total volume of inflow from the open boundary at t is Qt.   Define that l is the horizontal distance from the flooding edge to the seawall and h_1 is the water height from the bottom on the landside, then

$$l = \frac{h_1 L}{H}.$$ (7.9)

When the landside is completely flooded, we have

$$Qt = \frac{1}{2} l h_1 D = \frac{1}{2} \frac{h_1^2 L}{H} D,$$ (7.10)

so that

$$h_1 = \sqrt{\frac{2QH}{LD} t}.$$ (7.11)

The overtopping height (h), which is defined as the depth from the reference level, can be determined as

$$h = -(H - \sqrt{\frac{2QH}{LD} t}).$$ (7.12)

The experiments were made for cases with Q = 1000 m^3/s, 800 m^3/s, 600 m^3/s, 400 m^3/s and 200 m^3/s. For each case, the model was run until $h_1 = H$ (or $h = 0$). The

comparison between the model-computed and analytical overtopping heights for all five cases is shown in Fig. 7.4. The model accurately matched the analytical solutions. The slight bias near $t = 0$ was due to time-dependent oscillations during the model initial ramp up period. .

This idealized experiment demonstrates that the dike-groyne algorithm is capable of predicting the volume-conservative overtopping process from the ocean side to the landside.
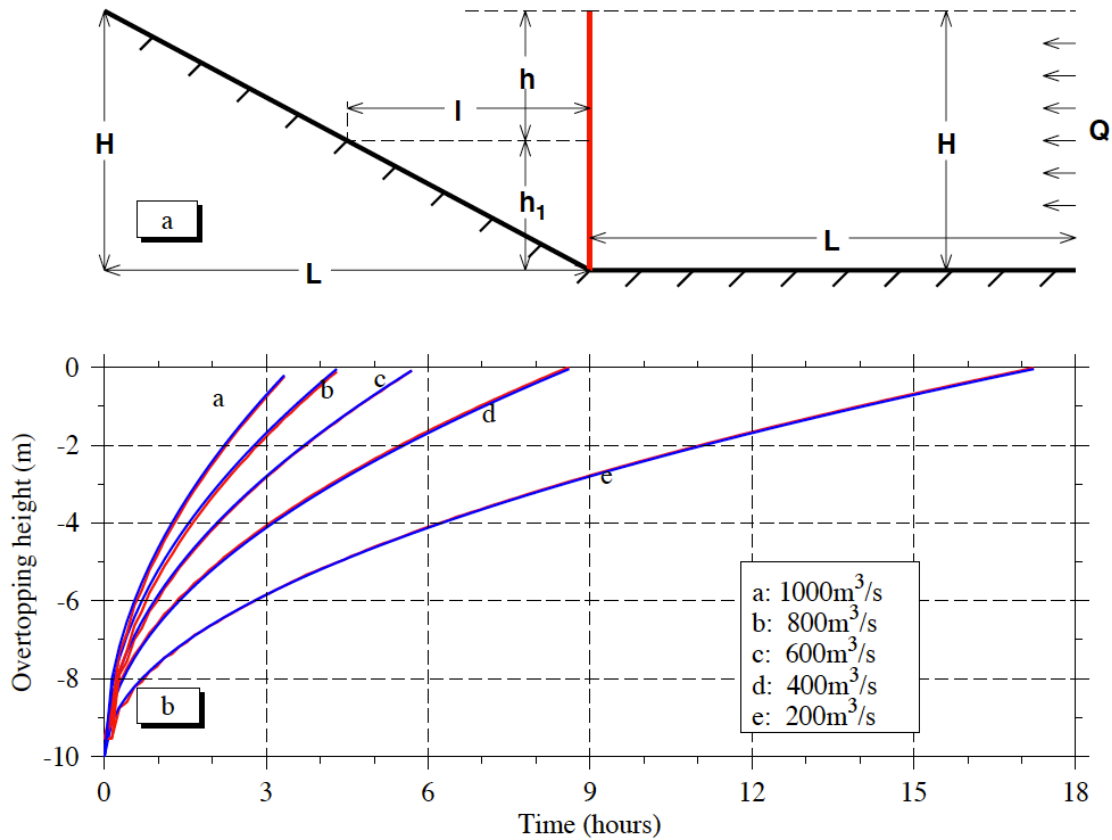
Figure 7.4: (a) Schematic of the model set up for the overtopping process experiments. (b) Model-data comparison of the overtopping depth . In panel a: H = height of the seawall; L = horizontal length of the land slope and ocean region; Q = water discharge rate at the open boundary; h = overtopping depth from the reference level; h_1 = overtopping height from the bottom; l = distance from the seawall to the flooded edge on the landside. In panel b: blue lines are the analytical solutions and red lines the model results.

Case II: Eddy formation in a fixed-bed flume. Yossef and Vriend (2011) conducted a laboratory experiment with aims at examining flow features in a fixed-bed flume (schematized as a straight river) with five groynes (Fig. 7.5). The experiments were made for the cases with emerged and submerged groynes. The results suggested that for a given inflow transport, groynes can produce a periodic flow fluctuation and the formation of multiple small-scale eddies between groynes. We repeated this laboratory experiment using FVCOM with the dike and groyne treatment module.
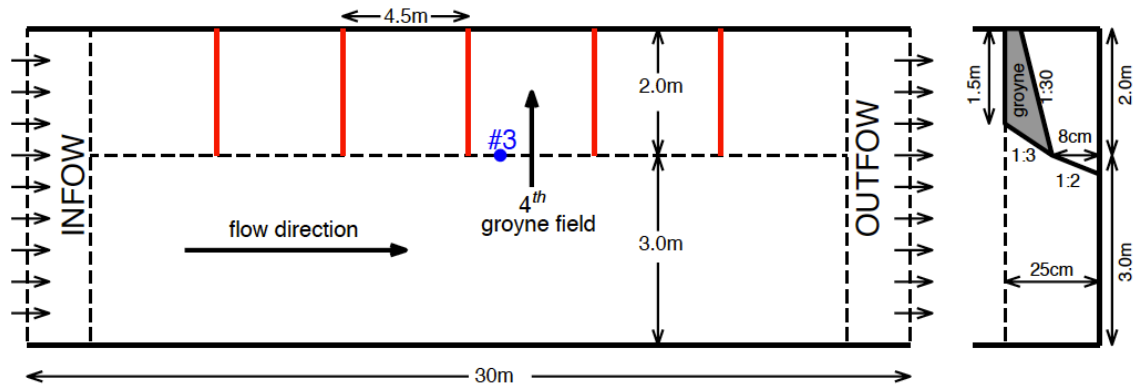
Figure 7.5: Plan view of the geometric structure and forcing setup used in the laboratory experiments by Yossef and Vriend (2011) and also used in our experiments. The right panel indicates the cross-sectional view. The gray region is the groyne, and the blue point #3 is the measurement site, which is over a 0.75 m distance from the third groyne tip. The data collecting depth is at 0.3 h of the whole water column.

Numerical experiments were made with the same configuration as laboratory experiments made by Yossef and Vriend (2011). The fixed-bed flume is constructed with x, y and z dimensions of 30 m in length, 5 m in width, and of 25 cm in height (Fig. 7.5). Five 2-m long groynes are attached on one side of the flume with a separation distance of 4.5 m. Groynes have a slope edge with a scale shown in the right side panel of Fig. 7.5. The region off groynes is defined as the main channel, which is 3 m in width. A constant and uniform water transport is specified as inflow on the left-side boundary and the same amount of water transport is specified as outflow on the right-side boundary. Three laboratory experiments were made in Yossef and Vriend (2011): one (Exp#1) for an emerged condition with water transport $Q = 0.248$ m^3/s and flow depth $H = 0.248$ m, two for submerged conditions with (Exp#2: $Q = 0.305$ m^3/s, $H = 0.310$ m) and (Exp#2: $Q = 0.381$ m^3/s, $H = 0.357$ m). Here we consider Exp#1 and Exp#2 for our model validation.

FVCOM was configured with non-overlapped triangular mesh with a uniform horizontal resolution of 5 cm. A total of 10 sigma layers were specified in the vertical, with a vertical resolution of ~2-3 cm in the main channel. The vertical and horizontal viscosities were set up to have the same Reynolds number of 6×10^4 in the main channel and 10^4 in the groyne region as estimated in the laboratory. The model was integrated for 1000 seconds, with inclusion of a 100-second ramp up time.
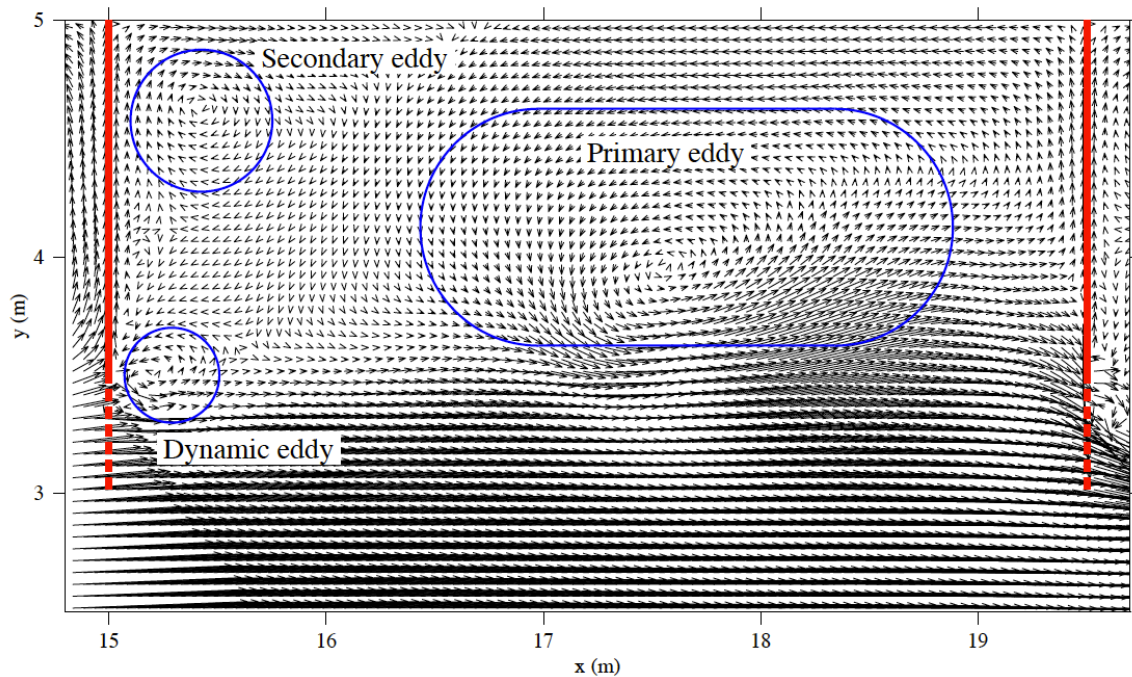
Figure 7.6: Snapshot of simulated flow patterns in 4th groyne field for Exp#1 under an emerged groyne condition. Solid red lines indicate emerged groynes, and dashed red lines indicate the submerged sloped edges of groynes. Blue cycles shows the locations of primary, secondary and dynamic eddies.

FVCOM is capable of reproducing the fluid features observed in the laboratory experiments made by Yossef and Vriend (2011). For the emerged groyne case, the laboratory experiment shows three types of eddies between groynes [see Fig. 7 in Yossef and Vriend (2011)]: 1) a cyclonic primary eddy in the downstream area between groynes, 2) an anti-cyclonic secondary eddy in the upper-left corner near the left side groyne, and 3) a cyclonic dynamic eddy at the slope edge of the left groyne. These three eddy features were captured in the FVCOM experiment (Fig. 7.6). The model results not only predict eddy structures, but also the spatial distribution of water exchanges between groynes. For

the submerged groyne case, a time series of velocity was recorded at point#3 (Fig. 7.5) in the laboratory experiment, which shows an oscillation with a period of ~30-35 seconds. The magnitudes and oscillation periods were captured in the FVCOM experiments (Fig. 7.7). The high-frequency variability recorded in the laboratory experiment was believed due to fluctuations in the sensor.

　Good agreement between model and laboratory experiments for emerged and submerged groyne cases demonstrate that the unstructured-grid dike-groyne treatment algorithm implemented in FVCOM is capable of capturing the essential physics in afluid flow system with dikes and groynes.
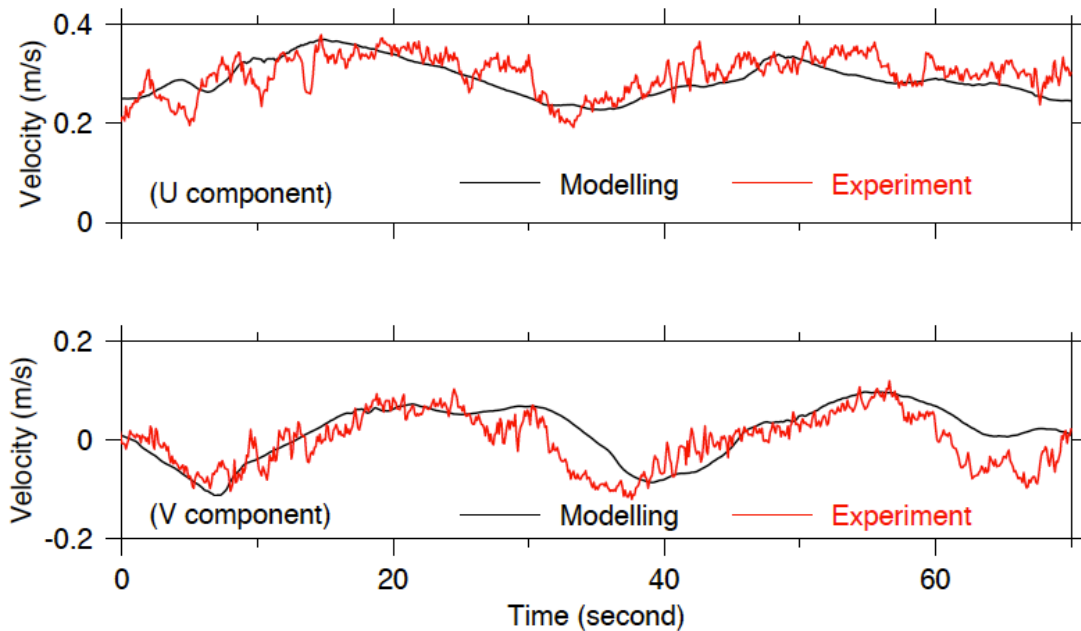
Figure 7.7: Model-data comparison of U- and V-components of the fluid velocity at point #3 for Exp#2 under a submerged groyne condition. Red lines indicate the time series of the measured velocity recorded in the laboratory and black lines are the model-computed velocity.

Chapter 8: FVCOM Sediment Module (FVCOM-SED)

In the early development of FVCOM, we implemented a simple sediment model to the Fortran 77 version when FVCOM was applied to study the Satilla River Estuary. That simple sediment model consisted of a 3-D passive tracer equation with the inclusion of a sinking-settling velocity and an elementary resuspension process. This model was not included in the parallel version of FVCOM because it was too simple to be a practical tool for the study of sediment transport processes in the coastal ocean.

G. Cowles has implemented a more complete 3-D sediment model module for FVCOM. This sediment model is based on the Community Model for Coastal Sediment Transport developed by the USGS and other researchers (see http://woodshole.er.usgs.gov/project-pages/sediment-transport/), which includes suspended sediment and bedload transport, layered bed dynamics based on the active layer concept, flux-limited solution of sediment setting, unlimited number of sediment classes and bed layers and cohesive sediment erosion/deposition algorithms. The implementation described herein is based on J. Warner's (USGS) coding for the Community Model in the Regional Ocean Modeling System (ROMS). A major effort was made to convert the structured-grid ROMS code to the unstructured-grid FVCOM code and to use the mass conservative finite-volume approach to calculate the sediment advection. By incorporating the same sediment dynamics and utilizing similar parameterizations and empirical constants, direct comparisons between FVCOM and ROMS results can be made. In addition, further development to the model can easily by implemented into FVCOM. This manual includes a brief description of the model dynamics, instructions on using the FVCOM-SED module, some details on the coding implementation, and results from a test case with suspended load only.

7.1. Governing Equations

The model includes transport of both suspended load and bedload. There exists a region near the bottom where there is no clear distinction between these two sources. In this implementation, however, the loads are computed separately and added together to

produce the total load. The suspended load model uses a concentration-based approach subject to the following evolution equation:

$$\frac{\partial C_i}{\partial t} + \frac{\partial u C_i}{\partial x} + \frac{\partial v C_i}{\partial y} + \frac{\partial (w - w_i) C_i}{\partial z} = \frac{\partial}{\partial x}(A_H \frac{\partial C_i}{\partial x}) + \frac{\partial}{\partial y}(A_H \frac{\partial C_i}{\partial y}) + \frac{\partial}{\partial z}(K_h \frac{\partial C_i}{\partial z}) \qquad (8.1)$$

where C_i is the concentration of sediment i, A_h is the horizontal eddy viscosity and K_h is the vertical eddy viscosity. The settling velocity w_i is prescribed by the user for each sediment type in the input parameter file. At the surface, a no-flux boundary condition is used for the sediment concentration:

$$K_h \frac{\partial C_i}{\partial z} = 0, \quad z = \zeta \qquad (8.2)$$

At the bottom, the sediment flux is the difference between deposition and erosion:

$$K_h \frac{\partial C_i}{\partial z} = E_i - D_i, \quad z = -H \qquad (8.3)$$

The erosion rate is calculated as

$$E_i = \Delta t Q_i (1 - P_b) F_{bi} \left(\frac{\tau_b}{\tau_{ci}} - 1 \right) \qquad (8.4)$$

where Q_i is the erosive flux, P_b is the bottom porosity, F_{bi} is the fraction of sediment i in the bottom, τ_b is the bottom shear stress, and τ_{ci} is the critical shear stress of sediment i.

Sediment is scoured when the local bottom shear stress reaches a critical user-defined value and is removed at a rate defined by the user. The resulting concentration profile is dependent on a balance between advection, vertical diffusion, introduction of new material through erosion, and loss of material from the water column through settling. A constant, user-defined sink rate w_i is used to model settling for each sediment type. The settling term (term 4 in the evolution equation) must be carefully calculated due to the sharp gradients in the concentration profile that can occur near the bottom. In this implementation, a flux-limited scheme is used for the settling equation. This scheme introduces anti-diffusion by means of a min-mod limiter and maintains second-order spatial accuracy away from extremes. The settling flux through the bottommost control volume is saved and recycled as the depositional flux D_i in the evolution equation bottom boundary condition. The bedload is treated using the Meyer-Peter and Müller scheme to calculate the local load. The transport is determined by calculating the divergence of the

local load. Users can select the empirical parameters of the Meyer-Peter and Müller scheme through the sediment input file.

8.2. A Simple Test Case

The test case described here corresponds to Case 1: Channel Flow of the USGS Community Model for Coastal Sediment Transport. This case tests the primary dynamics of suspended load in the sediment model including treatment of settling, erosion, vertical diffusion, and advection. Neither the bedload nor any specific dynamics of the bed strata are considered in this case. See the USGS web page provided above for specifics regarding the case setup. Fig. 8.1 shows the resulting sediment concentration field and Fig. 8.2 displays the steady state sediment concentration profile at x = 8000 m for both ROMS and FVCOM. The difference between the two profiles is quite small.
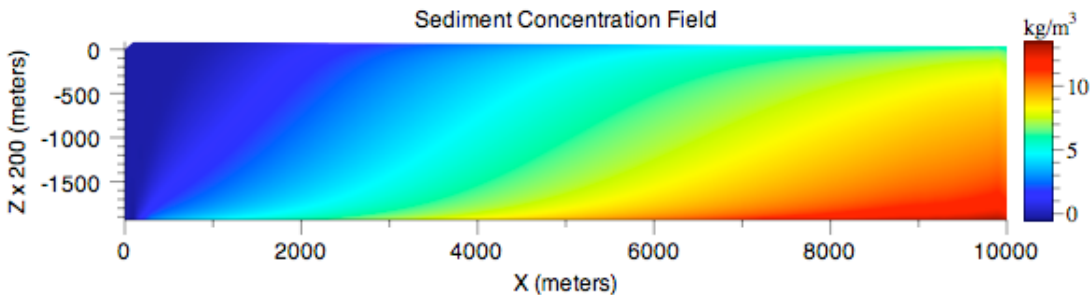
Fig. 8.1: The model-predicted sediment concentration for the test case 1.
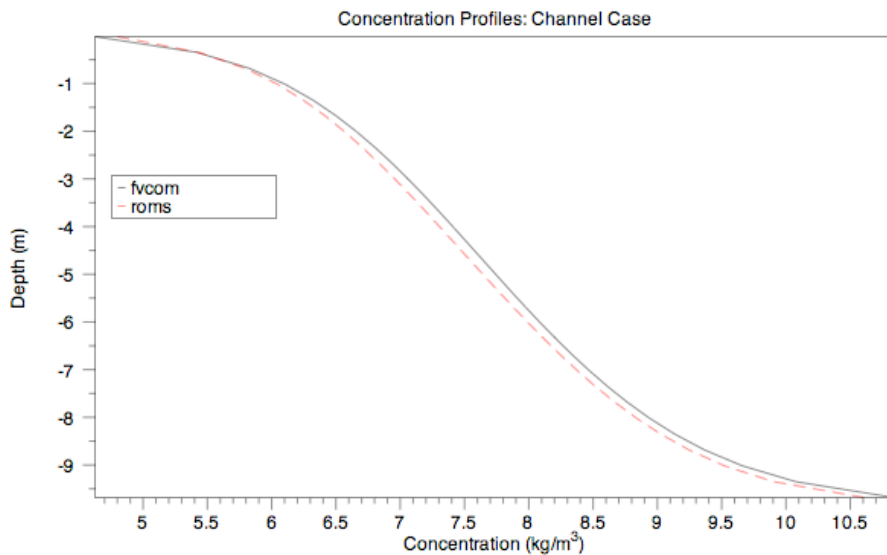
Fig. 8.2: A comparison of the FVCOM- and ROMs-computed steady state sediment concentration profiles at $x = 8000$ m for the test case 1.

Chapter 9: **FVCOM Surface Wave Module (FVCOM-SWAVE)**

9.1 Introduction

We have converted the structured-grid surface wave model SWAN (Simulating Waves Nearshore) to an unstructured-grid, finite-volume version under the FVCOM framework (Qi et al., 2009). This new unstructured-grid version of SWAN is named FVCOM-SWAVE. SWAN (Simulating WAves Nearshore) is the third-generation surface wave model developed originally by Booij et al. (1999) and improved through a team effort (SWAN Team, 2006a). This model considers the characteristics of surface waves in shallow water by solving the wave action balance equation with inclusion of dissipation from bottom friction, triad and quadruplet wave-wave interactions, and shallow water wave-breaking (SWAN Team, 2006b). SWAN has become one of the more popular surface wave models presently available and it is widely used for coastal ocean wave simulations, engineering applications and surface wave forecasts. SWAN is discretized using a curvilinear-structured grid and solved using fully implicit finite-difference algorithms. By application of a coarse-fine grid nesting approach, SWAN can be set up with variable grids in deep and shallow ocean regions to provide high quality simulations of surface waves in the nearshore region. A challenge for SWAN is to configure its curvilinear-structured grid to resolve realistic coastal geometry. Built on the same governing equations and control parameters, FVCOM-SWAVE provides an alternative option for the unstructured-grid model approach.

The implementation is made using the Flux-Corrected Transport (FCT) algorithm in frequency space, the implicit Crank-Nicolson method in directional space, and options of explicit or implicit second-order upwind finite-volume schemes in geographic space. FVCOM-SWAVE was developed for use in coastal ocean regions with complex irregular geometry. It has been validated for four idealized benchmark test problems that were used for SWAN validation. These problems are used to test numerical dispersion, wave-current interactions, wave propagation over a varying-bathymetry shallow water region, and the basic wave grow curves. Results were summarized and discussed in detail in Qi et al. (2009), which demonstrate that in the rectangular geometric domain, the second-

order finite-volume method used in FVCOM-SWAVE has the same accuracy as the third-order finite-difference method used in SWAN.

FVCOM-SWAVE has been fully coupled with FVCOM and FVCOM-SED. The coupled code has been validated for a tidal inlet test problem that was used for ROMS. FVCOM-SWAVE can be run with multi-nested domains. The nesting features two approaches, the first involves passing significant wave height and peak period data from one domain to the other and the second passes the wave spectrum data. The first approach is simple and runs more efficiently while the second approach ensures full dynamical transfer. A brief description of the discrete algorithms used in FVCOM-SWAVE and examples of validation results are given in this chapter.

9.2 Governing Equations and Discrete Algorithms

The governing equation of the wave action density spectrum can be written as

$$\frac{\partial N}{\partial t} + \nabla \cdot [(\vec{C}_g + \vec{V})N] + \frac{\partial C_\sigma N}{\partial \sigma} + \frac{\partial C_\theta N}{\partial \theta} = \frac{S_{tot}}{\sigma} \tag{9.1}$$

where N is the wave action density spectrum, t is the time, σ is the relative frequency, θ is the wave direction, C_σ and C_θ are the wave propagation velocities in spectral space (σ, θ), $\vec{C}_g = \partial \sigma / \partial \vec{k}$ is the group velocity, $\vec{k}$ is the wave number vector, $\vec{V}$ is the ambient water current vector, and $\nabla \cdot (\,)$ is the horizontal divergence operator in geographic space. In the Cartesian coordinates, $\nabla \cdot (\,) = \partial (\,) / \partial x + \partial (\,) / \partial y$, while in spherical coordinates, we denote λ as longitude and φ as latitude, implying $\nabla \cdot (\,) = \partial (\,) / \partial \lambda + \cos^{-1} \varphi \partial [\cos \varphi (\,)] / \partial \varphi$. S_{tot} is the source-sink term given as

$$S_{tot} = S_{in} + S_{nl3} + S_{nl4} + S_{ds,w} + S_{ds,b} + S_{ds,br} \tag{9.2}$$

where S_{in} is the function for the wind-induced wave growth, S_{nl3} is the nonlinear transfer of wave energy due to three-wave interactions, S_{nl4} is the nonlinear transfer of wave energy due to four-wave interactions, $S_{ds,w}$ is the wave decay due to white capping, $S_{ds,b}$ is the wave decay due to bottom friction, and $S_{ds,br}$ is the wave decay due to depth-induced wave breaking. Detailed descriptions of each of these terms are given in the SWAN technical manual (SWAN team, 2006b) and not included here.

Eq. (9.1) is discretized with four equations given as

$$\frac{N^{n+\frac{1}{4}} - N^n}{\Delta t} + \frac{\partial (C_\sigma N)}{\partial \sigma} = 0 \tag{9.3}$$

$$\frac{N^{n+\frac{2}{4}} - N^{n+\frac{1}{4}}}{\Delta t} + \frac{\partial (C_\theta N)}{\partial \theta} = 0 \tag{9.4}$$

$$\frac{N^{n+\frac{3}{4}} - N^{n+\frac{2}{4}}}{\Delta t} + \nabla \cdot [(\vec{C}_g + \vec{V})N] = 0 \tag{9.5}$$

$$\frac{N^{n+1} - N^{n+\frac{3}{4}}}{\Delta t} = \frac{S_{tot}}{\sigma} \tag{9.6}$$

where n denotes the nth time step, and Δt is the time interval for the numerical integration. Eqs. (9.3) and (9.4) describe the change of action density spectrum in spectral space. We used the Flux Corrected Transport method (FCT) (Boris and Book, 1973; Hsu et al., 2005] and the Crank-Nicolson method (Crank and Nicolson, 1947) to solve these two equations, respectively. Eq. (9.5) describes the propagation of the waves in geographic space. We adopted the FVCOM finite-volume method to solve it using either an explicit finite-volume upwind advection scheme or a semi-implicit finite-volume upwind advection scheme. Eq. (9.6) represents the growth, transfer and decay of the waves driven by the source terms. It is solved by a semi-implicit integration scheme as used in the WAM model (WAMDI Group, 1988) and WAVEWATCH III model (Tolman, 2002). A brief description of the discrete algorithms used to solve Eqs. (9.3)-(9.6) is given below.

Action density in frequency space. The FCT method, proposed first by Boris and Book (1973), is a conservative, positive discrete algorithm suitable for steep-gradient problems without dispersively generated oscillations. The discrete approach used in the FCT method consists of transport, anti-diffusion and correcting stages, as given by

$$N_{j_\sigma}^{n+1/4} = N_{j_\sigma}^* - (A_{j_\sigma+1/2}^* - A_{j_\sigma-1/2}^*), \tag{9.7}$$

where j_σ denotes the jth frequency and the resolution is specified as a "constant-relative-frequency" defined as $\Delta\sigma/\sigma$. $N_{j_\sigma}^*$ represents the action density at the transport stage calculated by

$$N_{j_\sigma}^* = N_{j_\sigma}^n - \frac{\Delta t}{\Delta\sigma}(\Phi_{j_\sigma+1/2}^1 - \Phi_{j_\sigma-1/2}^1) \tag{9.8}$$

where Φ is the flux defined as

$$\Phi^1_{j_\sigma+1/2} = N^n_{j_\sigma} \frac{C_{\sigma,j_\sigma+1} + |C_{\sigma,j_\sigma+1}|}{2} + N^n_{j_\sigma+1} \frac{C_{\sigma,j_\sigma+1} - |C_{\sigma,j_\sigma+1}|}{2} \qquad (9.9)$$

$$\Phi^1_{j_\sigma-1/2} = N^n_{j_\sigma-1} \frac{C_{\sigma,j_\sigma} + |C_{\sigma,j_\sigma}|}{2} + N^n_{j_\sigma} \frac{C_{\sigma,j_\sigma} - |C_{\sigma,j_\sigma}|}{2} \qquad (9.10)$$

and the superscript "1" denotes the first stage. $A^*_{j_\sigma+1/2}$ and $A^*_{j_\sigma-1/2}$ are limited anti-diffusion fluxes defined as

$$A^*_{j_\sigma+\frac{1}{2}} =$$

$$\mathrm{sgn}(A_{j_\sigma+\frac{1}{2}})\max\left\{0, \min\left[\left|A_{j_\sigma+\frac{1}{2}}\right|, \mathrm{sgn}(A_{j_\sigma+\frac{1}{2}})(N^*_{j_\sigma+2} - N^*_{j_\sigma+1}), \mathrm{sgn}(A_{j_\sigma+\frac{1}{2}})(N^*_{j_\sigma} - N^*_{j_\sigma-1})\right]\right\} \qquad (9.11)$$

$$A^*_{j_\sigma-\frac{1}{2}} =$$

$$\mathrm{sgn}(A_{j_\sigma-\frac{1}{2}})\max\left\{0, \min\left[\left|A_{j_\sigma-\frac{1}{2}}\right|, \mathrm{sgn}(A_{j_\sigma-\frac{1}{2}})(N^*_{j_\sigma+1} - N^*_{j_\sigma}), \mathrm{sgn}(A_{j_\sigma-\frac{1}{2}})(N^*_{j_\sigma-1} - N^*_{j_\sigma-2})\right]\right\} \qquad (9.12)$$

where

$$A_{j_\sigma+1/2} = \frac{\Delta t}{\Delta\sigma}\left(\Phi^2_{j_\sigma+1/2} - \Phi^1_{j_\sigma+1/2}\right); \quad A_{j_\sigma-1/2} = \frac{\Delta t}{\Delta\sigma}\left(\Phi^2_{j_\sigma-1/2} - \Phi^1_{j_\sigma-1/2}\right); \qquad (9.13)$$

$$\Phi^2_{j_\sigma+1/2} = N^n_{j_\sigma+1} \frac{C_{\sigma,j_\sigma+1} + C_{\sigma,j_\sigma}}{2}; \quad \Phi^2_{j_\sigma-1/2} = N^n_{j_\sigma} \frac{C_{\sigma,j_\sigma-1} + C_{\sigma,j_\sigma}}{2}; \qquad (9.14)$$

and superscript "2" denotes the second stage and $\mathrm{sgn}(A_{j_\sigma+1/2}) = \begin{cases} 1, & if\ A_{j_\sigma+1/2} \geq 0 \\ -1, & if\ A_{j_\sigma+1/2} < 0 \end{cases}$.

Action density in directional space. The action density at the $(n+2/4)^{th}$ time step in wave directional space is calculated using a second-order accurate implicit Crank-Nicolson scheme (Crank and Nicolson, 1947) given by

$$N^{n+2/4}_{j_\theta} = N^{n+1/4}_{j_\theta}$$
$$+ \alpha\frac{\Delta t}{2\Delta\theta}\left[(C_\theta N)^{n+2/4}_{j_\theta-1} - (C_\theta N)^{n+2/4}_{j_\theta+1}\right] - (1-\alpha)\frac{\Delta t}{2\Delta\theta}\left[(C_\theta N)^{n+1/4}_{j_\theta+1} - (C_\theta N)^{n+1/4}_{j_\theta-1}\right] \qquad (9.15)$$

where j_θ denotes the jth direction interval and the resolution is specified by $\Delta\theta$. α is a weighting factor with a default value of 0.5.

Action density in geographic space. Eq. (9.5) is solved numerically using the unstructured-grid finite-volume approach implemented in FVCOM (Chen et al., 2003a;

Chen et al., 2006d) for both Cartesian and spherical coordinates. The flux form of Eq. (5) in a control volume shown in Fig. 9.1 can be written as

$$N^{n+3/4} = N^{n+2/4} - \frac{\Delta t}{\Omega} \sum_{i=1}^{l_n} C_{n,i} N_{l_i} \Delta l_i .$$

(9.16)

Here, Ω is the area of the control volume indicated by the dark shaded area in Fig. 9.1, l_i ($i=1,l_n$) is the perimeter of Ω, l_n is the number of edges of Ω, and $C_{n,i}$ is the component of $\vec{C}_g + \vec{V}$ normal to l_i. Eq. (9.16) is solved by either the explicit upwind scheme or semi-
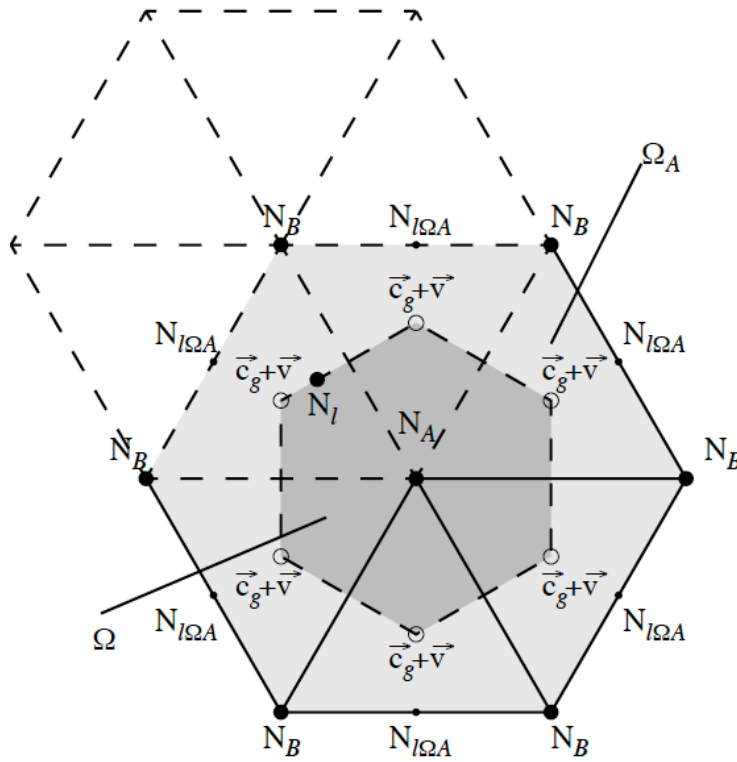
Fig. 9.1: Schematic of the unstructured triangular grid used for geographical spatial discretization in FVCOM- SWAVE. Definitions of variables are provided in the text.

implicit upwind scheme. Detailed descriptions of these two solvers are given in Qi et al. (2010); the calculation of the action density at the edge of the control volume is summarized here. In the explicit approach,

$$N_l = \begin{cases} N_A^{n+2/4} + \dfrac{\Delta r_A}{\Omega_A} \displaystyle\sum_{j=1}^{l_{\Omega_A,n}} N_{l_{\Omega_A,j}}^{n+2/4} \Delta l_{\Omega_A,j} & \text{for } C_n > 0 \\[3ex] N_B^{n+2/4} + \dfrac{\Delta r_B}{\Omega_B} \displaystyle\sum_{j=1}^{l_{\Omega_B,n}} N_{l_{\Omega_A}}^{n+2/4} \Delta l_{\Omega_A,j} & \text{for } C_n < 0 \end{cases}, \qquad (9.17)$$

where respectively, N_A^n and N_B^n are the nth time step action densities at nodes of A and B, Ω_A (light dashed area in Fig. 9.1) and Ω_B are the total area of triangles with central nodes at A and B, $l_{\Omega_A,j}$ ($j=1$, $l_{\Omega_A,n}$) and $l_{\Omega_B,j}$ ($j=1$, $l_{\Omega_B,n}$) are the perimeters of Ω_A and Ω_B, $l_{\Omega_A,n}$, $l_{\Omega_B,n}$ are the number of edges of Ω_A and Ω_B, and Δr_A and Δr_B are the distances from node A and node B to the centroids of triangles connected to nodes A and B. $C_n > 0$ refers to the outward direction of the control volume. This is the second-order approximate advection scheme used to solve the tracer equation in FVCOM.

In the semi-implicit approach, we have

$$N_l = \begin{cases} N_A^{n+3/4} + \dfrac{\Delta r_A}{\Omega_A} \displaystyle\sum_{j=1}^{l_{\Omega_A,n}} N_{l_{\Omega_A,j}}^{n+2/4} \Delta l_{\Omega_A,j} & \text{for } C_n > 0 \\[3ex] N_B^{n+3/4} + \dfrac{\Delta r_B}{\Omega_B} \displaystyle\sum_{j=1}^{l_{\Omega_B,n}} N_{l_{\Omega_A}}^{n+2/4} \Delta l_{\Omega_A,j} & \text{for } C_n < 0 \end{cases}. \qquad (9.18)$$

Substituting Eq. (9.18) into Eq. (9.16) results in a 2-D asymmetric and diagonally dominant matrix with a stencil equal to the sum of the surroundingnode points contained in a control volume. It can be solved efficiently using a scalable sparse matrix solver library (PETSc) (Balay et al., 2007) implemented with a high performance pre-conditional HYPRE software library (HYPRE Team, 2001). This method of sparse matrix solution was implemented to solve the Poisson equation for the non-hydrostatic version of FVCOM (Lai et al., 2010a).

Action density (N^{n+1}) related to source terms. Eq. (9.6) is solved using the same second-order, semi-implicit, centered-difference scheme as is implemented in WAM and WAVEWATCH-III. This is

$$N^{n+1} = N^{n+3/4} + \frac{\Delta t}{2\sigma}(S^{n+1} + S^n) \qquad (9.19)$$

where S^{n+1} and N^{n+1} are nonlinearly coupled to each other. A detailed description of this algorithm is given by the WAMDI Group (1988) and Tolman (2002).

9.3 Examples of Validation Experiment Results

Four idealized benchmark tests, which were used for SWAN validations, were repeated using FVCOM-SWAVE. These tests were designed to investigate the numerical diffusion of the discrete schemes and to examine the properties of wave-current interactions and wave propagation over varying shallow water topography. We also did standard growth curve analysis in idealized fetch-limited cases, following the SWAMP Group (1985), with a constant wind blowing seaward off a long straight coastline (as shown in Booij et al., 1999). A detailed description of the model validation results for
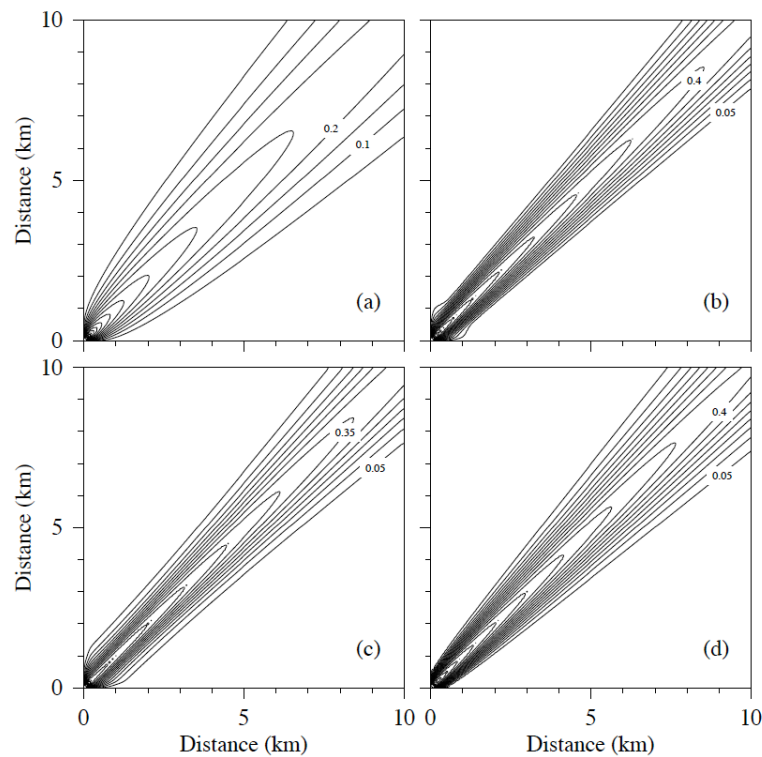
Fig. 9.2: Spatial distributions of the significant wave height (m) for a harmonic wave propagating along a diagonal line in a square computational domain. (*a*) SWAN-BSBT; (*b*) SWAN-SORDUP; (*c*) SWAN-SL; and (*d*) FVCOM-SWAVE using SORDUP.

these four cases is given in Qi et al. (2009), and we select one case here to illustrate the advection accuracy of FVCOM-SWAVE with comparison to SWAN.

Consider a harmonic, long-crested wave propagating through a gap into a square computational domain with dimensions 10 km × 10 km in deep water (Fig. 9.2). The

open gap is located in the lower left corner of the domain, so that the wave propagates along the diagonal at an angle of 45º with respect to the positive *x*-axis (*x* is the east-west coordinate which is positive in the eastward direction). This harmonic wave is simulated using a Gaussian-shaped frequency spectrum with a peak frequency of 0.1 Hz, a standard deviation of 0.01 Hz and a resolution defined as 3% of the relative frequency. The significant wave height at the gap is 1.0 m, and the long crest of the wave is calculated using an assumed $\cos^{500}(\theta)$ directional distribution.

The computational domain is tessellated with a square grid for SWAN and with right triangles for FVCOM-SWAVE. The right triangles are constructed by dividing each square along its diagonal line. The horizontal resolution for SWAN is 100 m, which is the same for FVCOM-SWAVE, where the horizontal resolution is defined using the shortest edge of a computational cell. The resolution in directional space is 0.5º. Here, the time step in SWAN and FVCOM-SWAVE is specified to maintain stability.

For this case, SWAN was run using three different discrete schemes: a) the first-order, backward space and backward time scheme for stationary waves (BSBT), b) the second-order upwind iteration scheme for stationary waves (SORDUP) (Rogers et al., 2002), and c) the third-order Stelling-Lendrertse scheme. FVCOM-SWAVE was run using both explicit and implicit second-order finite-volume upwind schemes.

The SWAN and FVCOM-SWAVE significant wave height distributions are compared in Fig. 9.2. The width of the spreading of the significant wave height field is used as an index for numerical diffusion. In general, the SORDUP result has the least numerical diffusion, whereas BSBT gives the largest numerical diffusion. Respectively, at distances of 1 and 5 km in the *x*- and *y*- axes away from the source, the width of the wave height field is 1.75 and 2.5 km for the first order BSBT, 1.0 and 1.5 km for the second order SORDUP, 1.5 and 1.75 km for the third-order Stelling-Lendretse scheme, and 0.75 and 1.75 km for FVCOM-SWAVE. If only the non-stationary numerical schemes are considered, we find that the second-order finite-volume upwind scheme used in FVCOM-SWAVE can reach the same level of numerical accuracy as the third-order Stelling-Lendretse scheme, and can exceed this level of accuracy in the region close to the gap. In this special case, the effective horizontal resolution is the same for both SWAN and FVCOM-SWAVE, because the computational cells have equivalent area.

Because BSBT is a first-order approximation, it is not surprising that this scheme (used in SWAN) generates the largest numerical diffusion. In the most current version of SWAN, this scheme is only used in cells connected to solid boundaries for both stationary and non-stationary waves.

9.4 Coupling of FVCOM, FVCOM-SWAVE and FVCOM-SED

FVCOM has included a full coupling of FVCOM, FVCOM-SWAVE and FVCOM-SED. The schematic of the fully current-wave-sediment coupling is shown in Fig. 9.3 and described in detail in the Ph.D. thesis of Wu et al. (2011). Coupling is approached through radiation stress, bottom boundary layer, surface stress, and morphology. The radiation stresses are added into the momentum equations of FVCOM to include the wave-driven motions. The bottom boundary layer (BBL) code with inclusion of the wave-current-sediment interaction developed by Warner et al. (2008) was converted into an unstructured-grid finite-volume version using the FVCOM framework and then implemented into FVCOM. At the sea surface, the sea surface roughness used to calculate the wind stress is calculated

Fig. 9.3: Schematic of the coupling between FVCOM circulation model (OCEAN), FVCOM-SWAVE and FVCOM-SED with inclusion of the wave-current-sediment bottom boundary layer model.

using the formulae described in Donelan et al. (1993) and given as

$$Z_o = 3.7 \times 10^{-5} \frac{U_{10}^2}{g} (\frac{U_{10}}{C_p})^{0.9}$$
(9.20)

where C_p is the phase velocity of the peak frequency, U_{10} is the 10-m height wind speed, and U_{10}/C_p stands for wave age. The drag coefficient is

$$C_d = (\kappa / \ln(10/z_o))^2$$
(9.21)

where κ = 0.41 is the von Karman constant. In the coupled FVCOM system, FVCOM-SWAVE runs with the input of the water velocity, sea surface elevation and bathymetry change. The morphological changes are accounted for by equating the bottom-boundary condition of the vertical velocity to the rate of change of

Fig.9.4: FVCOM grids and bathymetric distribution used for the inlet test problem.
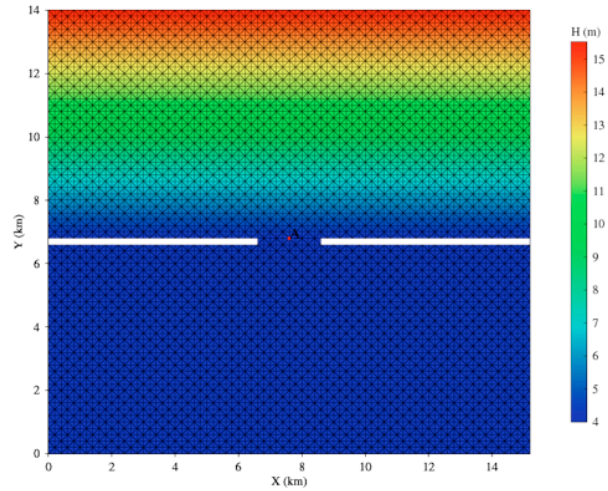
elevation of the sea floor. This method guarantees the mass conservation. A morphological scale factor is adopted from Roelvink (2006).

The coupled system is validated for an idealized tidally-driven coastal inlet. The
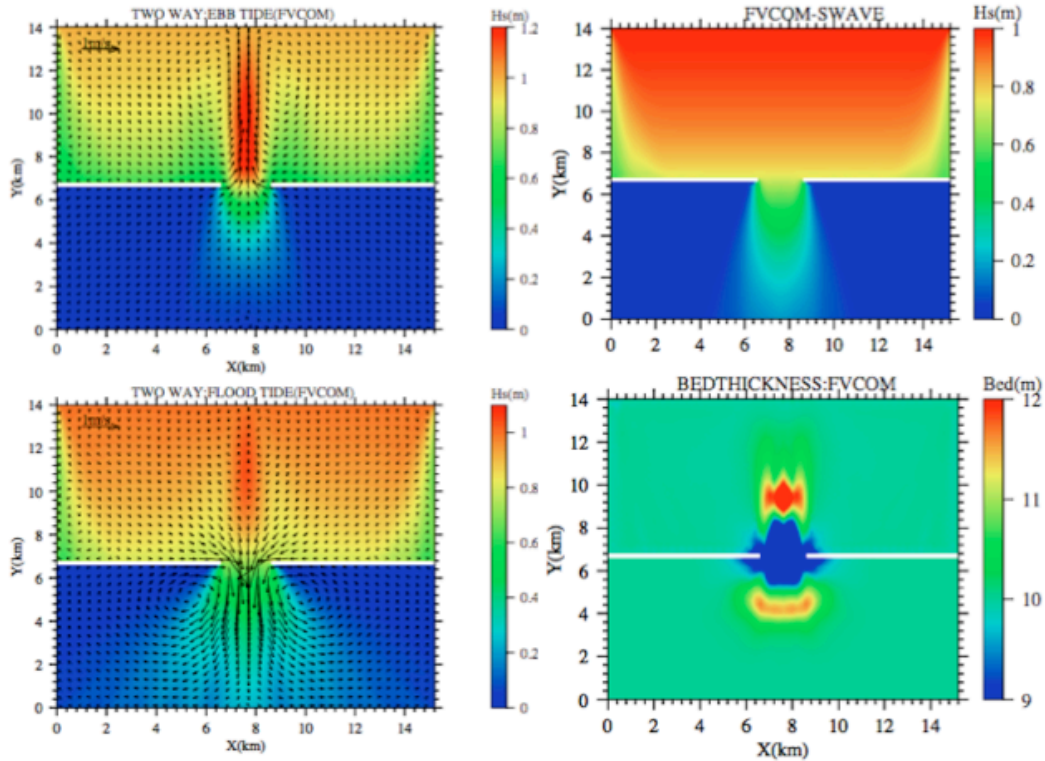
Fig.9.5: Distribution of currents at maximum ebb and flood tides overlapped the significant wave height (left panels), SWAVE-predicted significant wave height, and bed thickness.

geometrical domain of this inlet features a semi-enclosed rectangular basin with a width of 15 km and a length of 14 km. The initial water depth is specified to be 15 m at the open boundary (y = 0) and linearly decreases to 4 m at the northern end (y = 15 km). The initial bed thickness is specified to be 10 m everywhere. A wall is placed across the middle line of the basin, with a 2-km wide water exit at the center. A 1-m amplitude M_2 tide is specified at the open boundary (y = 0 km). Surface waves with 1-m height and 10-s period are also imposed at the open boundary and propagate southward towards the coast.   The validation results are described and discussed in Wu et al. (2011) and snapshots of model-predicted currents at ebb and flood tides, significant wave height (predicted by FVCOM-SWAVE only) and bed thickness are shown in Fig. 9.5.

Chapter 10: FVCOM Ice Model (UG-CICE)

An effort has been made to convert the structure-grid Community Ice CodE (CICE) to the unstructured-grid finite-volume version (UG-CICE) and couple it to FVCOM. Following the same finite-volume algorithms, UG-CICE discretizes the integral forms of the governing equations and solves them numerically by flux calculations over non-overlapped triangular meshes. This finite-volume approach is better at guaranteeing mass conservation in both individual control volumes and the entire computational domain (Chen et al., 2003a, 2007; Huang et al, 2008). In view of this technical approach, UG-CICE combines the best attributes of finite-difference methods for simple discrete computational efficiency and finite-element methods for geometric flexibility, which extends the capability of CICE to resolve ice dynamics in coastal regions characterized with irregular coastlines, numerous islands, barriers and narrow water passages.

UG-CICE was developed as a part of G. Gao's Ph.D. thesis research. It was originally implemented in FVCOM 2.7 and then upgraded to FVCOM 3.1.6 or up as an ice module coupled fully with ocean processes. The discretization algorithms and validations of this model are presented in detail in Gao et al. (2011). A brief description is given below.

10.1 Thermodynamic Processes

In CICE, the sea ice is assumed as a mixture of individual constituents with different thickness (Thorndike et al., 1975). A sea ice pack is mixtures of open water, thin first-year ice, thick multiyear ice, and thick pressure ridges, and these thickness constituents vary with similar thermal and mechanical forcing. A detailed description of thermodynamics processes was given in Hunke and Lipscomb (2006). A brief description of key processes is repeated below.

Solar irradiance. The solar radiation reaching the snow/ice (F_{SW}) consists of two parts: 1) the direct solar radiation (F_{SWdr}) that penetrates through the atmosphere and 2) the diffuse radiation (F_{SWdf}) scattered partially in the atmosphere. These two incoming shortwave parts can be separated into two irradiative quantities: a) visible radiation with wavelengths < 700 nm and b) near-infrared radiation with wavelengths > 700 nm. The snow and ice albedos used in CICE are determined based on the observations from the

SHEBA field experiment (Curry et al, 2001, Hunke and Lipscomb, 2006). The net heat flux at the top boundary of the ice from the atmosphere is defined as F_{TOP}, which is given as

$$F_{TOP} = F_{SH} + F_{LH} + F_{LW\downarrow} + F_{LW\uparrow} + \left(1 - i_0\right)F_{SW}$$ (10.1)

where F_{SH} is the sensible heat flux, F_{LH} is the latent heat flux, $F_{LW\downarrow}$ is the incoming longwave flux, $F_{LW\uparrow}$ is the outgoing longwave flux, F_{SW} is the incident shortwave flux at the snow/ice surface, and i_0 is the fraction of the absorbed shortwave flux that penetrates into the ice. The formulations of the heat flux components described above were described in details by Briegleb (1992), Rosati and Miyakoda (1988), and Hunke and Lipscomb (2006).

Ice and ocean heat flux exchange. Ice melting occurs due to the heat from the ocean when the water temperature (T_w) is higher than the freezing temperature (T_f). The melting potential heat (F_{max}) stored in the ocean is a function of the difference between water and freezing temperatures, oceanic mixing layer thickness (h_o), and seawater heat capacity (c_w) given as

$$F_{\max} = -c_w h_0 \rho_w \left(T_w - T_f\right)$$ (10.2)

where ρ_w is the density of seawater. The freezing temperature is also a function of water salinity given as $-\mu S$ (S is the seawater salinity and $\mu = 0.054$°C/psu). The ice can melt from its bottom and sides. The bottom melting is controlled by the net heat flux (F_{BOT}) between the ice and ocean, which can be estimated by

$$F_{BOT} = -\rho_w c_w c_h u^* (T_w - T_f)$$ (10.3)

where $c_h = 0.006$ is a heat transfer coefficient (McPhee, 1992) and u^* is the friction velocity that is equal to the square root of the kinematic stress at the ice-ocean interface (Maykut and Untersteiner 1971; Maykut, 1978; Maykut ad Perovich, 1987). Lateral melting is controlled by the interfacial heat exchange between the ocean and side boundaries of the ice, which is parameterized using the vertically averaged lateral melting rate given as

$$M_a = m_1 \left(T_w - T_f\right)^{m_2}$$ (10.4)

where $m_1 = 1.6\times10^{-6}$ m s^{-1} deg$^{-m_2}$ and $m_2 = 1.36$ (Josberger and Martin, 1981; Perovich,

1983; Maykut and Perovich, 1987). The role of lateral melting relative to the floe melting can be estimated using the so-called friction of lateral melting (R_{SID}) that is proportional to the ratio of the lateral melting rate to the effective floe diameter (D_F) (Steele, 1992) given as

$$R_{SID} = \left| \frac{M_a \pi}{\alpha \cdot D_F} \right| \tag{10.5}$$

where $\alpha = 0.66$ is a non-dimensional empirical constant. When $\pi/\alpha D_F \ll 1$, the role of lateral melting is negligible, and the large floes melt from the top and bottom surfaces. The lateral melting heat flux (F_{SID}) can be estimated by

$$F_{SID} = E_{tot} R_{SID} \tag{10.6}$$

where E_{tot} is the melting energy. In CICE, the sum of bottom and lateral melting heat fluxes must be smaller than F_{max}.

Temperature changes of snow and ice.   The temperatures within the ice is determined by the thermal conduction and absorption of penetrating shortwave irradiance given as

$$\rho_i c_i \frac{\partial T_i}{\partial t} = \frac{\partial}{\partial z} \left(k_i \frac{\partial T}{\partial z} \right) - \frac{\partial}{\partial z} \left[I_0 \exp(-\kappa_i z) \right] \tag{10.7}$$

where subscript "i" represents the ith layer of the ice, $\rho_i = 917 kg/m^3$ is the sea ice density, c_i is the specific heat of sea ice, k_i is the thermal conductivity of sea ice, I_0 is the intensity of solar irradiance at the top surface, and κ_i is the extinction coefficient by ice attenuation of solar irradiance. This equation can also be applied to snow by changing the subscript "i" to "s". The boundary conditions at the top surface are specified for four general cases: 1) snow present and no melting, 2) snow present with melting, 3) snow absent and no melting, and 4) snow absent and no melting. A detailed description of each case can be found in the CICE User Manual (Hunke and Lipscomb, 2006).

Open-water ice growth and evolution. When the enthalpy of new ice is known, melting at the top surface is controlled by the difference between the net heat flux from the atmosphere to the ocean and the conductive flux from the top surface to the ice interior. Similarly, growth and melting at the bottom ice surface is controlled by the difference between the conductive heat flux at the bottom surface and the net heat flux

between ice and ocean. If the latent heat flux is negative (i.e., the latent heat is transferred from the ice to the atmosphere), snow or snow-free ice will sublimate at the top surface. If the latent heat flux is positive, vapor from the atmosphere is deposited at the surface as snow or ice. For positive latent heat fluxes, the deposited snow or ice is assumed to have the same enthalpy as the existing surface layer. After growth and melting, the various ice layers no longer have equal thicknesses. The layer interfaces will be changed by conserving energy.

Snow-ice conversion. The snow layer overlying the sea ice can be thick enough in a heavy snow condition to depress the snow-ice interface below the sea level and cause the seawater to flood the snow. The snow is converted to ice when

$$\rho_s h_s > (\rho_w - \rho_i) h_i .$$ (10.8)

The snow base lies below sea level as

$$h^* = \frac{h_s \rho_s - (\rho_w - \rho_i) h_i}{\rho_s} > 0 .$$ (10.9)

In this case, the snow base will be raised to sea level by converting some snow to ice following mass conservation,

$$\delta h_s = \frac{-\rho_i h^*}{\rho_w} \quad \text{and} \quad \delta h_i = \frac{\rho_s h^*}{\rho_w}$$ (10.10)

10.2 Ice Dynamics Processes

UG-CICE uses the same governing equations as CICE and is coded for both spherical and Cartesian coordinates. The full nonlinear governing equation of the ice movement in the Cartesian coordinate system, for example, is given as

$$\begin{cases} m\dfrac{\partial u}{\partial t} + m\left(u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} \right) - mfv = \dfrac{\partial \sigma_{1j}}{\partial x_j} - mg\dfrac{\partial \zeta}{\partial x} + \tau_{ax} + \tau_{wx} \\[4mm] m\dfrac{\partial v}{\partial t} + m\left(u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} \right) + mfu = \dfrac{\partial \sigma_{2j}}{\partial x_j} - mg\dfrac{\partial \zeta}{\partial y} + \tau_{ay} + \tau_{wy} \end{cases}$$ (10.11)

where m is the combined mass of ice and snow per unit area, u and v are the x- and y-components of the ice velocity, g is gravity, ζ is the sea surface elevation, σ_{ij} is the internal stress tensor with subscripts of 1, 2 and i, j (in a range from 1 to 2) representing

x-(noted as "1") and y-(noted as "2") components, and (τ_{ax}, τ_{ay}) and (τ_{wx}, τ_{wy}) are the x- and y-components of sea surface wind and water stresses, respectively. The vector forms of wind and water stresses are formulated as

$$\vec{\tau}_a = cC_a\rho_a|\vec{u}_a|(\vec{u}_a\cos\varphi + k\times\vec{u}_a\sin\varphi) \tag{10.12}$$

$$\vec{\tau}_w = cC_w\rho_w|\vec{u}_w - \vec{u}_s|\left[(\vec{u}_w - \vec{u})\cos\theta + \vec{k}\times(\vec{u}_w - \vec{u}_s)\sin\theta\right] \tag{10.13}$$

where c is the ice concentration ranging from 0 to 1, $\vec{u}_s$ is the surface velocity vector, C is the drag coefficient, ρ is the density, and ϕ and θ are the air and water turning angles, respectively. The subscripts "a" and "w" represent "air" and "water", respectively. In spherical coordinates, the x (eastward) and y (northward) axes are defined as

$$x = r\cos\phi(\lambda - \lambda_0) \text{ and } y = r(\phi - \phi_0) \tag{10.14}$$

where r is the earth's radius, λ is longitude, φ is latitude, and λ_0 and φ_0 are the reference longitude and latitude, respectively. Following the coordinate module in FVCOM (Chen, et al., 2006), eqn. (10.11) can be easily converted into the spherical coordinate system.

UG-CICE considers the Elastic-Viscous-Plastic (EVP) dynamics (Hunke and Dukowicz, 1997) in which σ_{ij} is derived from the equation

$$\frac{1}{E}\frac{\partial\sigma_{ij}}{\partial t} + \frac{1}{2\eta}\sigma_{ij} + \frac{\eta-\zeta}{4\eta\zeta}\sigma_{kk}\delta_{ij} + \frac{P}{4\zeta}\delta_{ij} = \dot{\in}_{ij} \tag{10.15}$$

where E is the elastic parameter defined by Young's modulus given as $E = \hat{\zeta}/T$, $\hat{\zeta}$ is the bulk viscosity, T is a damping timescale for elastic waves, η is the shear viscosity, P is the ice strength, $\in_{ij}$ is the ice strain rate, k represents i or j, and δ_{ij} is the Kronecker function defined as $\delta_{ij}=1$ for $i=j$ and 0 for $i\neq j$. $\in_{ij}$ has the form

$$\dot{\in}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{10.16}$$

P can be estimated by one of two empirical formulas derived by Hibler (1979) or Rothrock (1975) and Lipscomb et al. (2007). In Hibler (1979),

$$P = P^*h\exp[-C^*(1-c)], \tag{10.17}$$

where $P^* = 2.75\times10^4 N/m^2$, h is the mean ice thickness, and $C^* = 20$ is an empirical constant. In Lipscomb et al. (2007),

$$P = C_f C_p \beta \sum_{n=1}^{N_c} [-ap_n h_n^2 + \frac{ap_n}{k_n}(H_{\min}^2 + 2\lambda H_{\min} + 2\lambda^2)] \,. \tag{10.18}$$

where C_f is an empirical parameter for frictional energy dissipation, $C_p = (g/2)$ $(\rho_i/\rho_w)(\rho_w - \rho_i)$, ρ_i is the ice density, $\beta = R_{\text{tot}}/R_{\text{net}} > 1$, $R_{\text{tot}} = \sum_{n=1}^{N_c} r_n$, r_n is the ridging rate, N_c is the total number of ice categories, n is the number of each category from 1 to N_c, R_{net} is the net rate of area loss for the ice pack, ap_n is the thickness distribution of the ice participating in ridging, h_n is the ice thickness for the category n, k_n is the ratio of the mean ridge thickness to the thickness of ridging ice, $H_{\min} = 2h_n$, and λ is an empirical e-folding scale.

Ignoring the local time change term in eq. (10.15), UG-CICE also can be simplified for the Viscous-Plastic (VP) rheology. In this case, σ_{ij} is a function of the ice strain rate and strength by the constitutive law (Hibler, 1979; Zhang and Hibler, 1997).

The ice transport equation satisfies the conservation law by which the thickness distribution function remains unchanged following the ice current trajectory. Define that $\hat{g}(\vec{x}, h, t)$ is the thickness distribution function containing the ice area, ice volume, snow volume, ice energy, snow energy and area-weighted surface temperature, the transport equation for $\hat{g}$ can be written as the standard advection equation

$$\frac{\partial \hat{g}}{\partial t} = -\nabla \cdot (\hat{g}\vec{u}) \,. \tag{10.19}$$

10.3 Discretization

UG-CICE retains the same thermodynamics equations as CICE. Since the thermal processes in CICE are 1-D (vertical) features, all discrete equations and algorithms can be directly adopted in UG-CICE. A major modification was made to change the discrete algorithms for the ice momentum equations to an unstructured-grid, finite-volume method following the FVCOM framework (Gao et al., 2011).

Eqs. (10.11)-(10.19) are discretized in the horizontal using a set of non-overlapped unstructured triangular meshes that subdivide the model domain. A triangle is comprised of three nodes; a centroid and three sides (Figure 10.1), on which u and v are placed at centroids and scalar variables like $\hat{g}$ are placed at nodes. u and v at centroids are

calculated based on the net flux through the three sides of that triangle (called the momentum control element: MCE), while scalar variables at each node are determined by the net flux through the sections linked to centroids and the middle point of the sideline in the surrounding triangles (called the tracer control element: TCE).
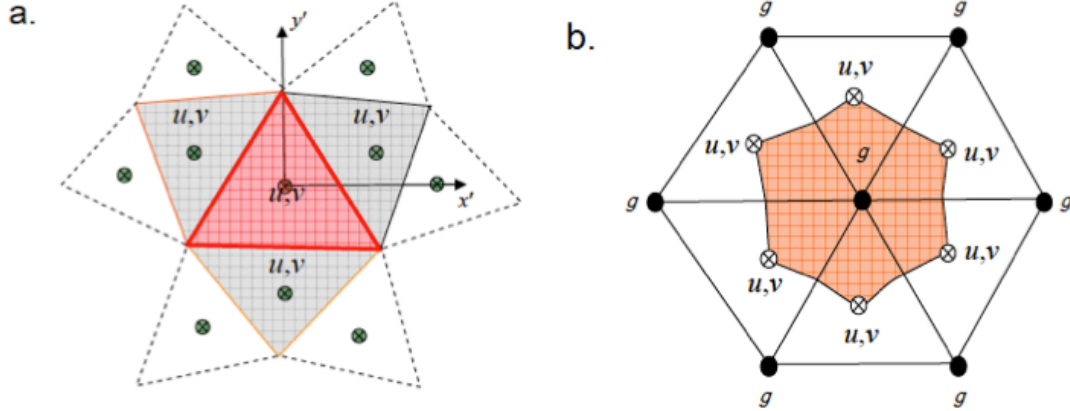
Figure 10.1: Schematic of the unstructured triangular grid used for geographical spatial discretization in UG-CICE (a: for the ice momentum equation; b: for the ice transport equation).

Integrating eqs. (10.11) and (10.15) over individual MCE area gives

$$\frac{\partial u}{\partial t} =$$
$$-\frac{1}{\Omega}[\oint_{s'} uv_n \, ds' + \iint_\Omega fv \, dx \, dy + \frac{1}{m}\oint_{s'}(-\sigma_{11} \, dy + \sigma_{12} dx) - g\oint_{s'} H_o \, dy + \iint_\Omega \frac{\tau_{ax} + \tau_{wx}}{m} dx dy] \tag{10.20}$$

$$\frac{\partial v}{\partial t} =$$
$$-\frac{1}{\Omega}[\oint_{s'} vv_n \, ds' - \iint_\Omega fu \, dx \, dy + \frac{1}{m}\oint_{s'}(-\sigma_{12} \, dy + \sigma_{22} dx) - g\oint_{s'} H_o \, dx + \iint_\Omega \frac{\tau_{ay} + \tau_{wy}}{m} dx dy] \tag{10.21}$$

$$\frac{\partial \sigma_1}{\partial t} + \frac{\sigma_1}{2T} + \frac{P}{2T} = \frac{P}{2T\Delta}D_D \tag{10.22}$$

$$\frac{\partial \sigma_2}{\partial t} + \frac{e^2\sigma_2}{2T} = \frac{P}{2T\Delta}D_T \tag{10.23}$$

$$\frac{\partial \sigma_{12}}{\partial t} + \frac{e^2\sigma_{12}}{2T} = \frac{P}{4T\Delta}D_S \tag{10.24}$$

where v_n is the velocity component normal to the sides of the triangle and s' is the closed

trajectory comprised of the three sides. $\sigma_1 = \sigma_{11} + \sigma_{22}$, $\sigma_2 = \sigma_{11} - \sigma_{22}$, $D_D = \in_{11} + \in_{22}$,

$D_T = \in_{11} - \in_{22}$, $D_S = 2\in_{12}$, $\Delta = \left[D_D^2 + \frac{1}{e^2}(D_T^2 + D_S^2) \right]^{1/2}$, and $e=2$ is the ratio of major to

minor axis of the elliptical yield curve for the principal components of the stress. In the EVP system, the internal stress is a function of the time- and space-satisfying tensor equations in eqs. (10.22)-(10.24), which requires a shorter time step to resolve elastic waves. Because the ice strain rates in these three equations are related to the ice velocity, eqs. (10.20)-(10.24) need to be integrated at the same time step. In the thermodynamics and ice transport time step (Δt), eqs. (10.20)-(10.24) are integrated at the subcycling time step of $\Delta t_e = \Delta t / N_d$, where $N_d=120$ is the default sub-integration number in both UG-CICE and CICE.

Eqs. (10.22)-(10.24) are discretized in a semi-implicit form given as

$$\begin{cases} \dfrac{\sigma_1^{\hat{n}+1} - \sigma_1^{\hat{n}}}{\Delta t_e} + \dfrac{\sigma_1^{\hat{n}+1}}{2T} + \dfrac{P}{2T} = \dfrac{P}{2T\Delta^n} D_D^{\hat{n}} \\[3mm] \dfrac{\sigma_2^{\hat{n}+1} - \sigma_2^{\hat{n}}}{\Delta t_e} + \dfrac{e^2\sigma_2^{\hat{n}+1}}{2T} = \dfrac{P}{2T\Delta^n} D_T^{\hat{n}} \\[3mm] \dfrac{\sigma_{12}^{\hat{n}+1} - \sigma_{12}^{\hat{n}}}{\Delta t_e} + \dfrac{e^2\sigma_{12}^{\hat{n}+1}}{2T} = \dfrac{P}{4T\Delta^n} D_S^{\hat{n}} \end{cases} \qquad (10.25)$$

and

$$\sigma_{11}^{\hat{n}+1} = 0.5(\sigma_1^{\hat{n}+1} + \sigma_2^{\hat{n}+1}) \quad \text{and} \quad \sigma_{22}^{\hat{n}+1} = 0.5(\sigma_1^{\hat{n}+1} - \sigma_2^{\hat{n}+1}). \qquad (10.26)$$

We adopted the second-order upwind scheme in FVCOM to solve eqs. (10.20)-(10.24). This finite-volume algorithm has been well described in Kobayashi (1999) and Chen et al. (2003a, 2006d) and a brief description is given here. Let R_u and R_v represent all the terms on the right hand side of the u and v eqs. (10.20)-(10.21), respectively, and superscript $\hat{n}$ represents the $\hat{n}^{\text{th}}$ time step within N_d step integration. In R_u and R_v, the internal stress terms are expressed implicitly at the $(\hat{n}+1)^{\text{th}}$ time step. Eqs. (10.20)-(10.21) with an implicit form of internal stresses are integrated numerically from the $\hat{n}^{\text{th}}$ time step to $(\hat{n}+1)^{\text{th}}$ time step using the modified fourth-order Runge-Kutta time-stepping scheme with second-order accuracy (Chen et al., 2003a). The integration procedure is given as

$$
\begin{cases}
u_R^0 = u^{\hat{n}}; \, v_R^0 = v^{\hat{n}}; \, R_u^0 = R_u^{\hat{n}}; \, R_v^0 = R_v^{\hat{n}}; \\[2mm]
u_R^k = u_R^0 - \alpha^k \dfrac{\Delta t_e R_u^{k-1}}{4\Omega}; \, v_R^k = v_R^0 - \alpha^k \dfrac{\Delta t_e R_v^{k-1}}{4\Omega}; \quad , \\[2mm]
u^{\hat{n}+1} = u_R^4; \, v^{\hat{n}+1} = v_R^4
\end{cases}
\tag{10.27}
$$

where k =1,2,3,4 and $(\alpha^1, \alpha^2, \alpha^3, \alpha^4)$ = (1/4, 1/3,1/2, 1). Ω is the triangular area of the MCE where u and v are located. After the ice velocity is integrated from eq. (10.16) at a subcycling step (Δt_e), the divergence terms, strain rates and viscosity can be updated for the next subcycling integration of eq. (10.25). The ice mass m, ice strength P and external wind and ocean stress do not change in the thermodynamics time step (Δt).

The ice transport eq. (10.19) is calculated in the integral form using the second-order upwind scheme in the form of

$$
\frac{\partial \hat{g}}{\partial t} = -\iint_{\Omega_{\hat{g}}} \nabla \cdot (\hat{g}\bar{u}) dx dy = -\oint_{l_{\Omega_{\hat{g}}}} \hat{g} v_n ds
\tag{10.28}
$$

where $\Omega_{\hat{g}}$ is the area of a TCE, the normal velocity v_n at the boundary of a TCE is given at each triangular centroid, $\hat{g}$ at the boundary is calculated by linear interpolation from the upwind control volume shown in Figure 10.1 where $\hat{g} = \hat{g}_t + \dfrac{\partial \hat{g}}{\partial x} \Delta x + \dfrac{\partial \hat{g}}{\partial y} \Delta y$;

$\dfrac{\partial \hat{g}}{\partial x} = 1/\hat{\Omega}_{\hat{g}} \oint \hat{g}\, dy$; $\dfrac{\partial \hat{g}}{\partial y} = 1/\hat{\Omega}_{\hat{g}} \oint \hat{g}\, dx$ where $\hat{\Omega}_{\hat{g}}$ is the area of the control volume in the upwind direction. A detailed explanation of the second-order upwind advection scheme used in the tracer equation calculation is given in Chen et al. (2006d).

10.4 Coupling of UG-CICE and FVCOM

The implementation of UG-CICE to FVCOM is at the ice-ocean interface with ice mass, ice stress, and heat exchange. Since UG-CICE was designed to have the same horizontal grid structure as FVCOM, coupling these two models is straightforward. Salt was treated as a conservative mass. FVCOM is a free-surface model and in the ice-free ocean, the salt flux is zero at the sea surface. Net precipitation minus evaporation (*P-E*) at

the surface provides freshwater into the ocean, which changes the salt concentration and the sea surface vertical velocity but not the salt flux at the air-sea interface. In the ice-covered ocean, the local change of the ice mass per unit area in UG-CICE and FVCOM is added to the kinematic condition of vertical velocity at the ice-ocean interface. In the ice model, the ice-ocean salt flux is calculated with an assumption of a constant reference salinity of 4 psu in ice. The salt flux at the ice-ocean interface in UG-CICE and FVCOM considers the variation of the total salinity change related to the ice volume change.

With ice, the surface boundary conditions for FVCOM's momentum equations are changed to

$$\frac{\partial}{\partial z}(K_m \frac{\partial u}{\partial z}) = 1/\rho_w [c\tau_{wx} + (1-c)\tau_{ax}] \tag{10.28}$$

$$\frac{\partial}{\partial z}(K_m \frac{\partial v}{\partial z}) = 1/\rho_w [c\tau_{wy} + (1-c)\tau_{ay}] \tag{10.29}$$

where c is the ice concentration and τ_{wx} and τ_{wy} is the x and y components of the ice-ocean interfacial stress. The kinematic surface condition plus the mass flux of fresh water is

$$w = \frac{\partial \zeta}{\partial t} + u\frac{\partial \zeta}{\partial x} + v\frac{\partial \zeta}{\partial y} + \frac{(P-E)}{\rho_w} + \frac{\partial m}{\rho_i \partial t} \tag{10.30}$$

where P and E are the precipitation and evaporation rates over an open area, and m is mass of ice per unit area.

If the ice is present, the ocean surface temperature is fixed to the freezing temperature ($T_f = -\mu S_0$) that is related to the surface salinity (Dirichlet boundary condition). The ice growth and melting is directly related to the heat potential (referring to freezing temperature). In the temperature equation, the net oceanic heat flux (F_{Ocn}) at the sea surface is determined by

$$F_{Ocn} = (1-c)F_{air} \tag{10.31}$$

where F_{air} is the heat flux from the atmosphere to the ocean. The shortwave irradiance

(F_{SW0}) is the sum of the shortwave irradiance of the open area (F_{SWa}) and shortwave irradiance penetrated through ice (F_{SWi}) given as

$$F_{SW0} = (1-c)F_{SWa} + cF_{SWi} \tag{10.32}$$

and the temperature flux condition at the sea surface is specified as

$$\frac{\partial T}{\partial z} = \frac{1}{\rho_w c_p K_h}\left[F_{Ocn} - F_{SW0} + cF_{OI} + (1-c)F_{frz}\right] \tag{10.33}$$

where F_{OI} is the net heat flux between the ocean and ice that is determined in ice simulation and F_{frz} is the heat gained when sea ice grows over open water.

The surface boundary conditions for salinity is:

$$\frac{\partial s}{\partial z} = \frac{Q_{salt}}{K_h \rho_w} \tag{10.34}$$

$$Q_{salt} = (S_0 - S_i)\frac{\partial m_i}{\partial t} + S_0\frac{\partial m_s}{\partial t} \tag{10.35}$$

where S_i is the ice salinity (4 psu), S_0 is the first level ocean salinity, and m_s and m_i are the masses of snow and ice per unit area.

10.5 UG-CICE Validations

UG-CICE has been validated for three idealized benchmark test problems used for CICE validation. The idealized problems were designed to test the capability of UG-CICE to resolve the EVP and VP ice dynamics, ridging scheme, numerical instability, and ice-current interaction in high- and low-ice concentration regimes, respectively. The validation results are summarized in detail in Gao et al. (2011). For the same rectangular domain, UG-CICE configured with an unstructured triangular grid is capable of reproducing the results of CICE. The finite-volume algorithm used in UG-CICE provides a better resolution of the sharp velocity change in the transition zone.

Gao et al. (2011) also applied UG-CICE to simulate the seasonal variability of sea ice in the Arctic Ocean. Driven by "climatologic" meteorological forcing, tides and river discharge, the model captured the seasonal variability of the sea ice concentration, coverage and drift velocity within the inter-annual variation range. An example of the model-data comparison of the ice concentration is shown in Figure 10.2. A detailed

explanation of these results is given in Gao et al. (2011).
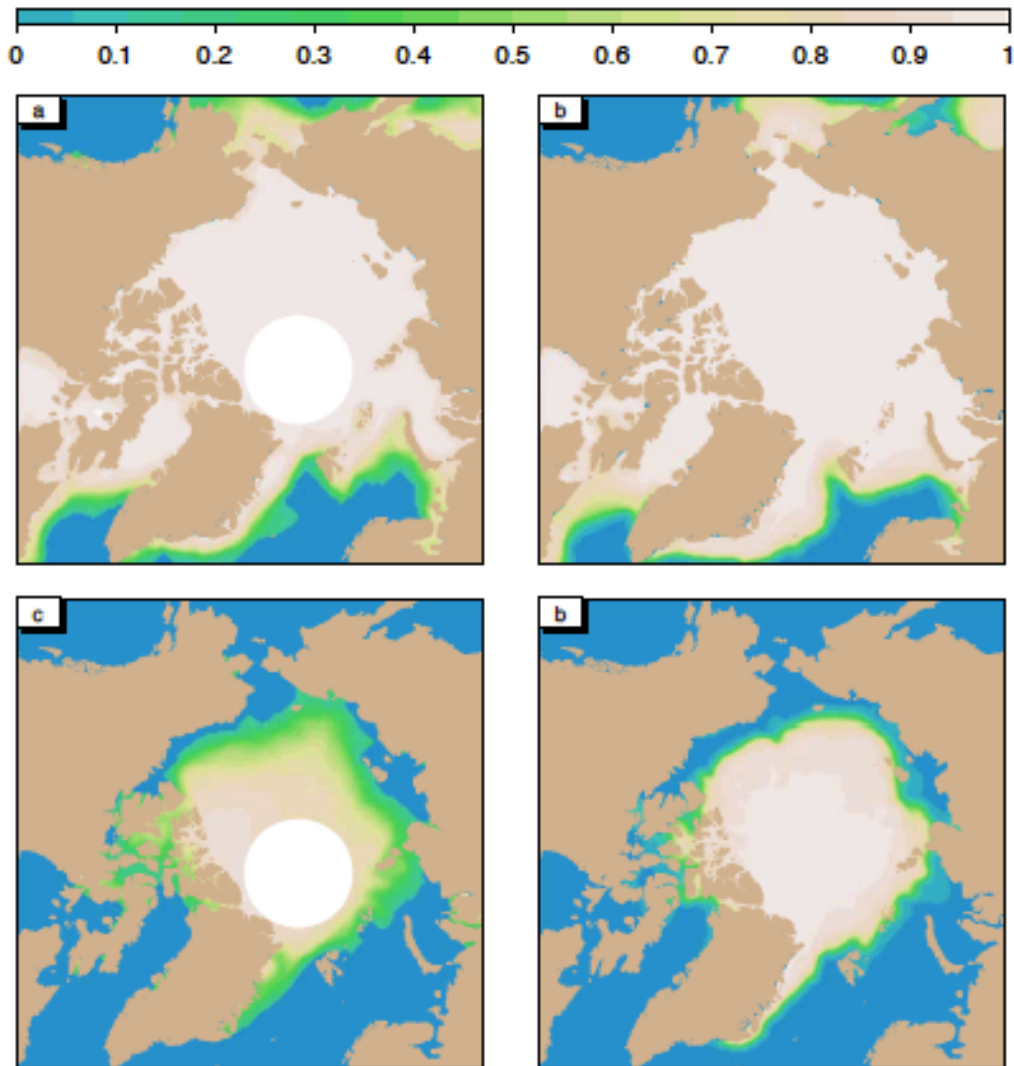
Figure 10.2: Comparison between distributions of UG-CICE-calculated (right panels) and satellite-derived (left panels) monthly ice concentration for March (upper panels) and September (lower panels).

Chapter 11: FVCOM Ecosystem Modules

11.1 Introduction

In 2005, we started implementing the Generalized Ecosystem Module (GEM) into FVCOM. The original plan was to convert the Harvard University structured-grid adjustable biological module code to an unstructured-grid version and add it into FVCOM as a biological module. At that time, FVCOM already had several biological modules, including: 1) a water quality model (coded for parallel execution), 2) a simple NPZ model, and 3) a 9-component NPZD model (coded for serial execution only). The idea was to build a GEM that would allow users to select either a pre-built biological model (such as NPZ, NPZD, etc) or construct their own biological model using the pre-defined pool of biological variables and parameterization functions. This module could be run simultaneously together with FVCOM with parallel execution (so-called "online" mode) or driven separately by FVCOM output ("offline" mode). This module would act as a platform that would allow users to examine the relative importance of different physical and biological processes under well-calibrated physical fields. Following the code structure of the Harvard General Biological Model, R. Tian modified the FVCOM existing biological code to build the first version of GEM. Because this version of GEM was coded in the main FVCOM programs but not as a module, the generality of this model made the coding difficult to follow, setup, run and debug. C. Chen reformulated the structure of GEM and designed an independent module of GEM. Q. Qi and R. Tian then worked together to construct the new GEM module following the mathematical equations and designs derived by C. Chen. The description of GEM in this chapter is based on C. Chen's notes that were used to construct the GEM for FVCOM.

We also implemented the following three water quality models into FVCOM: 1) EPA Quality Analysis Simulation Program (WASP) version (FVCOM-WQM), 2) UG-RCA, and and 3) UG-CE-QUAL-ICM. UG-RCA is an unstructured-grid version of the HydroQual RCA water quality model. UG-CE-QUAL-ICM is an unstructured-grid version of the Army Corps of Engineers structured-grid finite-volume eutrophication model. FVCOM-WQM is an online model and can run simultaneously with FVCOM.

UG-RCA and UG-CE-QUAL-ICM are offline models, which are driven using FVCOM output. A brief description of these three models is also given in this chapter.

11.2 General Ecosystem Module (GEM)

11.2.1. Flow Chart of GEM

The structure of the General Ecosystem Module was developed by dividing lower trophic food web processes into 7 state variable groups: 1) nutrients [$N(i)$, $i=1$, nn], 2) phytoplankton [$P(i)$, $i=1,np$], 3) zooplankton [$Z(i)$, $i=1$, nz], 4) detritus [$D(i)$, $i=1,nd$], 5) dissolved organic matter [$DOM(i)$, $i=1,nm$], 6) bacteria [$B(i)$, $i=1$, nb], and 7) auxiliary state variables [$Y(i)$, $i=1$, ny]. The flow chart of the transformation among these variables is shown in Fig. 11.1. We named this system "General Ecosystem Module (GEM)" to emphasize the point that this module provides a platform that allows users to build their own parallelized biological model from a

Fig. 11.1: Flow chart of the flexible biological module GEM.

discrete set of functions which are independent of the physical model. This module can be run either simultaneously with linkage to unstructured-grid (e.g., FVCOM) and structured-grid ocean models through the connection to the physical model dependent 3-D advection and diffusion variables or it can be run separately by itself in 1-D applications. This structure is adapted from the popular General Ocean Turbulence Model (GOTM) system that uses the same approach. We understand that the range of existing biological models is too vast and complex to try to encompass in a generalized way. In the GEM code, the biological module is an independent 1-D system that is self-maintained and upgraded without linking to a physical model. It is easy to extend the

GEM to a 3-D case by linking to the advection and diffusion modules of any physical model. It can be also converted to a Lagrangian-based biological model by linking it with the 3-D Lagrangian particle-tracking module.

The biological module in FVCOM includes point source input from rivers, nudging at lateral boundaries, air-sea interaction at the surface, and benthic fluxes at the bottom. Because these physical processes are already documented in the FVCOM manual and code, we will focus our description of the GEM here on the internal biological processes.

11.2.2. Equations and functions in GEM

11.2.2.1. Nutrients

Nutrients in the biological module include 1) ammonium (NH_4^+), nitrate (NO_3^-), nitrite (NO_2^-), phosphate (PO_4^-), and silicate ($Si(OH)_4$). We also include iron (Fe) in the nutrient equation because in many coastal and open ocean environments, it acts as a limiting nutrient in autotrophic processes. Two types of processes are included for nutrient field modification: 1) physical processes including external loading and advection, and 2)

P respiration: $\sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \gamma_{P_j} P_j$

Z passive respiration: $\sum_{j=1}^{nz} \Gamma_{N_i:C}^{Z_j} \gamma_{Z_j} Z_j$

Z active respiration: Z_{AR}

$\sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \gamma_{B_j} B_j$

$\sum_{k=1}^{nb} \mu_B(T)[\sum_{j=1}^{nm} \Gamma_{N_i:C}^{DOM_j}(1-e_j^{B_k})F_a]B_k$

Nutrient

N_i

$-\sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \mu_{P_j} P_j$

Bacteria uptake:

$-\sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \mu_{B_j}(T)(1-e_{NH_4}^{B_j})F_b B_j$

Remineralization $[\sum_{j=1}^{nn} \Gamma_{N_i:C}^{DOM_j} d_{N_j}^{R}]D_i$
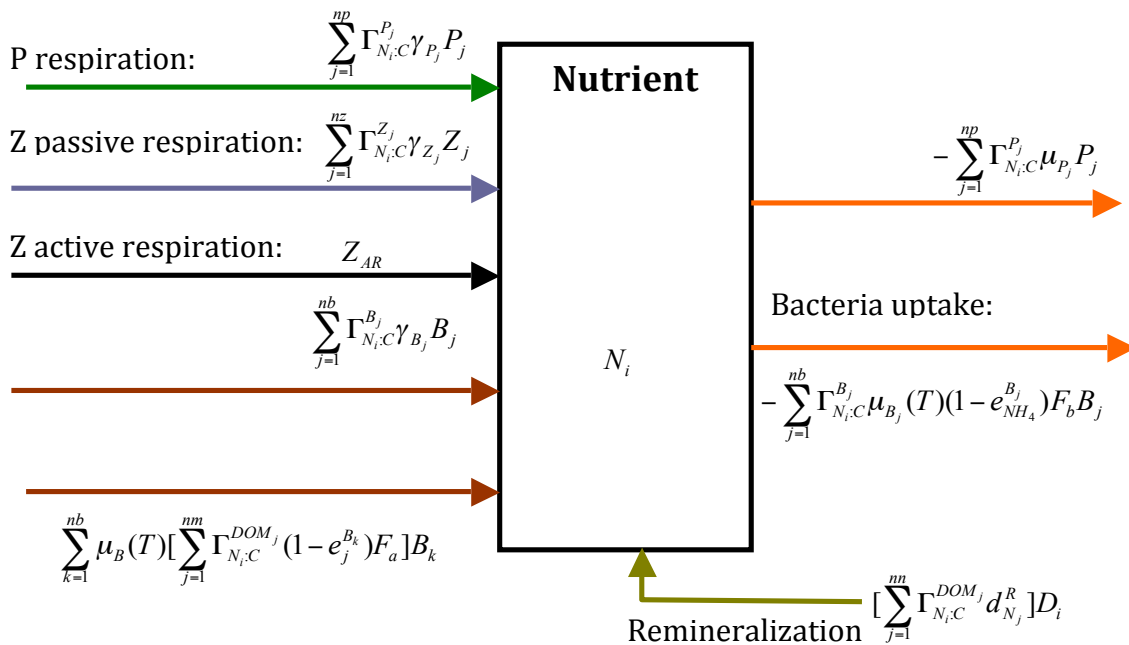
biological processes including internal regeneration resulting from respiration by phytoplankton, zooplankton, and bacteria, remineralization from detritus, and the loss of nutrients by phytoplankton and bacteria uptake. An inflow/outflow schematic of the biological processes is shown in Fig. 11.2.

To facilitate model use, carbon is used as the standard unit for all autotrophic and heterotrophic variables. For nutrients, the standard is the nutrient itself. Therefore, the ratio of a carbon to nutrient unit is defined to support conversion from one base to another. Define [N_i, i=1, nn] as the general form of the nutrient variables where i represents the ith nutrient variable and nn is the total number of nutrients included in the module. We can write the general form of the nutrient equation as

$$\frac{\partial N_i}{\partial t} = Loss\_N_i + Gain\_N_i + ADV\_N_i + HDIFF\_N_i + VDIFF\_N_i \qquad (11.1)$$

where $Loss\_N_i$ represents the ith nutrient uptake by phytoplankton and bacteria, $Gain\_N_i$ is the ith nutrient regeneration through respiration byphytoplankton, zooplankton, and bacteria, decomposition of detritus, and remineralization of DOM, ADV_N_i is the change of the ith nutrient due to horizontal and vertical advection, and $HDIFF\_N_i$ and $VDIFF\_N_i$ are the change of the ith nutrient due to horizontal and vertical diffusion, respectively. (Note that both ADV_N_i and $HDIFF\_N_i$ are not present in eq. (11.1) in 1-D (vertical) applications.) We also take nitrification into account in the nutrient equations for NO_3^-, NO_2^-, and NH_4^+. The mathematic expressions for $Loss\_N_i$ and $Gain\_N_i$ are

$$Loss\_N_i = P\_take + B\_uptake$$
$$= -\sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \mu_{P_j}(T) P_j - \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \mu_{B_j}(T)[1 - e_{NH_4}^{B_j}]F_b B_j \qquad (11.2)$$

and

$$Gain\_N_i = P\_respiration + Z\_passive\_respiration$$
$$+Z\_active\_respiration + B\_passive\_respiration + B\_active\_respiration \qquad (11.3)$$

where

$$P\_respiration = \sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \gamma_{P_j} P_j \qquad (11.4)$$

$$Z\_passive\_respiration = \sum_{j=1}^{nz} \Gamma_{N_i:C}^{Z_i} \gamma_{Z_j} Z_j \tag{11.5}$$

$$B\_passive\_respiration = \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \gamma_{B_j} B_j \tag{11.6}$$

$$Z\_active\ respiration = \sum_{k=1}^{nz} \{ \sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} [(1 - e_{P_j}(k) - \alpha_{P_j}^{D}(k) - \alpha_{P_j}(k)] G_{P_j}$$

$$+ \sum_{\substack{j=1 \\ j \neq i}}^{nz} [1 - e_{Z_j}(k) - \alpha_{Z_j}^{D}(k) - \alpha_{Z_j}(k)] + G_{Z_j} \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} [1 - e_{B_j}(k) - \alpha_{B_j}(k)] B_j \tag{11.7}$$

$$+ \sum_{j=1}^{nd} \Gamma_{N_i:C}^{D_j} [1 - e_{D_j} - \alpha_{D_j}(k) - \alpha_{B_j}^{D}(k)] D_j \}$$

$$B\_active\_respiration = \sum_{k=1}^{nb} \mu_{B_k}(T) [\sum_{j=1}^{nm} \Gamma_{N_i:C}^{DOM_j} (1 - e_j^{B_k}) F_a] B_k \tag{11.8}$$

$$D\_remineralization = [\sum_{j=1}^{nn} \Gamma_{N_i:C}^{DOM_j} d_{N_j}^{R}] D_j \tag{11.9}$$

$$P\_uptake = - \sum_{j=1}^{np} \Gamma_{N_i:C}^{P_j} \mu_{P_j}(T) P_j \tag{11.10}$$

$$B\_uptake = - \sum_{j=1}^{nb} \Gamma_{N_i:C}^{B_j} \mu_{B_j}(T) (1 - e_{NH_4}^{B_j}) F_b \tag{11.11}$$

where $\Gamma_{N_i:C}^{P_j}$, $\Gamma_{N_i:C}^{Z_j}$, $\Gamma_{N_i:C}^{B_j}$, $\Gamma_{N_i:C}^{D_j}$ and $\Gamma_{N_i:C}^{DOM_j}$ are the ratio of the ith nutrient N_i unit to carbon for the jth phytoplankton P_j, zooplankton Z_j, bacteria B_j, detritus D_j, and DOM_j (Table 11.1).

We also consider the nitrification from ammonium to nitrate through nitrifying bacteria. It has been recognized that nitrification is a process related to light. A simple empirical formula for the nitrification rate (Olson, 1981) is given here as an option:

$$Q_{NH_4 \to NO_3} = \begin{cases} 0 & I(t,z) \geq 0.1 I_{max} \\ \dfrac{0.1 I_{max} - I(t,z)}{0.1 I_{max}} & I(t,z) < 0.1 I_{max} \end{cases} \tag{11.12}$$

where I_{max} is the maximum light intensity at the surface, and I is the light intensity in the water column.

Table 11.1: Parameters used in the nutrient equations

Parameter	Definition	Unit
$\Gamma_{N_i:C}^{P_j}$	ratio of the ith nutrient N_i unit to carbon for the jth phytoplankton P_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_j}$	ratio of the ith nutrient N_i unit to carbon for the jth zooplankton Z_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{B_j}$	ratio of the ith nutrient N_i unit to carbon for the jth bacteria B_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{D_j}$	ratio of the ith nutrient N_i unit to carbon for the jth detritus D_j.	mmol N_i:mmol C
$\Gamma_{N_i:C}^{DOM_j}$	ratio of the ith nutrient N_i unit to carbon for the jth DOM_j	mmol N_i:mmol C

11.2.2.2. Phytoplankton

Phytoplankton growth is an autotrophic process that is dependent on light photosynthesis and nutrient limitation. The local change of the phytoplankton is controlled by phytoplankton growth, respiration, mortality, DOM exudation, sinking, zooplankton grazing, and physical advection and diffusion processes. The inflow/outflow chart of the ith phytoplankton species is shown in Fig.11.3. The equation for the ith phytoplankton species can be written as

$$\frac{\partial P_i}{\partial t} = \text{Growth of } P_i + \text{Respiration of } P_i + \text{Mortality of } P_i + \text{DOM exudation of } P_i$$

$$+ \text{ Sinking of } P_i + \text{Grazing of } P_i + \text{ADV\_}P_i + \text{HDIFF\_}P_i + \text{VDIFF\_}P_i \tag{11.13}$$

where

$$\text{Growth of } P_i = \Gamma_{C:N_i}^{P_i}(1 - d_i^{DOM})\mu_{P_i}(T)P_i \tag{11.14}$$

$$\text{Respiration of } P_i = -\gamma_{P_i} P_i \tag{11.15}$$

$$\text{Mortality of } P_i = -\varepsilon_{P_i} P_i^{m_i} \tag{11.16}$$

DOM exudation of P_i $= -d_{P_i} P_i$ (11.17)

Sinking of P_i $= -w_{P_i} \dfrac{\partial P_i}{\partial z}$ (11.18)

Grazing of P_i $= -\sum\limits_{j=1}^{nz} G_{P_i}(j)$ (11.19)
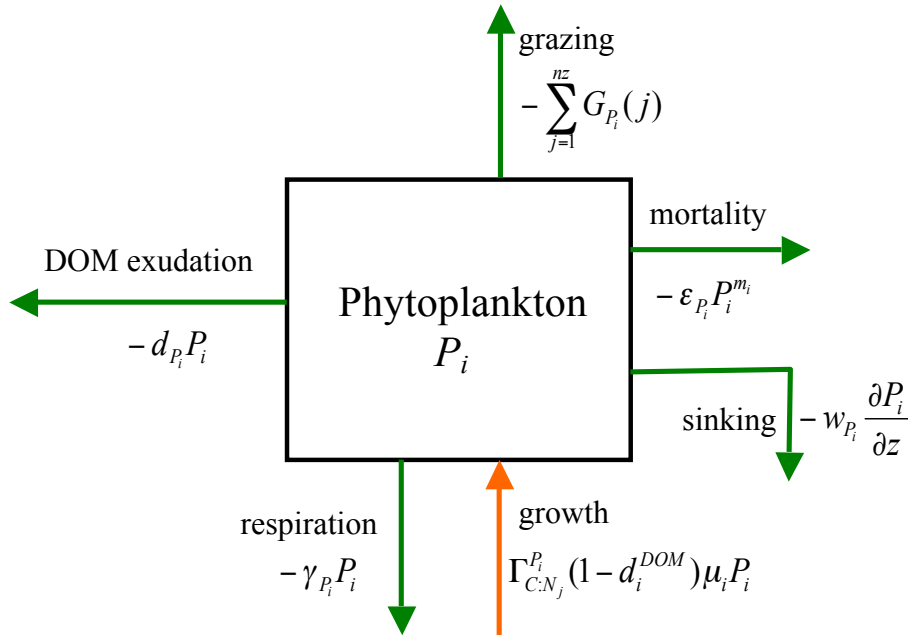
Fig. 11.3: Flow chart of the input and output in the phytoplankton species P_i equation.

and ADV_P_i, HDIFF_P_i, and VDIFF_P_i are the change of P_i due to advection, horizontal and vertical diffusion, respectively. $G_{P_i}(j)$ is the grazing loss by the jth zooplankton species. Definitions of parameters used in eqs. (11.13)-(11.19) are given in Table 11.2.

Table 11.2: Parameters used in the phytoplankton equations

Parameter	Definition	Unit
$\Gamma_{C:N_j}^{P_j}$	ratio of C to nitrogen	mmol C:mmol N
d_i^{DOM}	active DOM exudation	fraction
μ_{P_i}	gross growth rate of P_i	mmol C (mg Chl)$^{-1}$ s^{-1}

γ_{P_i}	respiration rate of P_i	s^{-1}
ε_{P_i}	mortality rate of P_i	s^{-1}
m_i	power of P_i in the mortality term	dimensionless
w_{P_i}	sinking velocity of P_i	$m\,s^{-1}$
G_{Pi}	zooplankton grazing rate	s^{-1}

There are many empirical methods to determine μ_{Pi}. Three factors are commonly considered in the estimation of μ_{Pi}: light intensity, water temperature, and nutrient limitation. In a multiple nutrient case, two popular empirical formulas for μ_i are given as

$$\mu_{iA} = \mu_i(T)\min[\mu_i(I),\mu_i(N_1),\mu_i(N_2),\cdots,\mu_i(N_{nn})] \tag{11.20}$$

$$\mu_{iB} = \mu_i(T)\mu_i(I)\min[\mu_i(N_1),\mu_i(N_2),\cdots,\mu_i(N_{nn})] \tag{11.21}$$

where $\mu_i(I)$ is the normalized light limitation function, $\mu_i(T)$ is the growth limitation due to the water temperature, and $\mu_i(N_i)$ is the nutrient limitation function due to the ith nutrient. To make the selection more flexible, we combine (11.20) and (11.21) together as follows:

$$\mu_i = \mu_{max}^{P_i}(\alpha\mu_{iA} + (1-\alpha)\mu_{iB}) \tag{11.22}$$

where α is a weighting coefficient ranging from 0 to 1 and $\mu_{max}^{P_i}$ is the maximum growth rate of phytoplankton P_i

$\mu_i(T)$ is usually expressed as the exponential function

$$\mu(T) = e^{-a_T|T-T_{opt}|} \tag{11.23}$$

where T_{opt} is the optimal water temperature at which the maximum growth rate has been measured and α_T is the exponential decay rate of $\mu_i(T)$ relative to the water temperature difference.

The selection of $\mu_i(I)$ is empirical, which relies on the best fit of the relationship between the growth rate and light intensity. To make the code more general, we include

13 expressions for $\mu_i(I)$ here in Table 11.3. These empirical formulas are added into the code in the form of a FORTRAN function to make it easy for users to select one of the existing formulas or add their own one.

Table 11.3: **Functions for $\mu_i(I)$ included in the FVCOM GEM**

Function Name	Equation	References
EXP_LIGHT	$e^{-k|z|}$	Franks et al. (1986)
SL62_LIGHT	$\dfrac{I}{I_{opt}} e^{\left(1-\frac{I}{I_{opt}}\right)}$	Steele (1962)
MM_LIGHT	$\dfrac{\alpha_I I}{K_I + \alpha_I I}$	Baly (1935); Tamiya et al. (1985); Caperon and Meyer (1972); Kiefer and Mitchell (1983)
LB_LIGHT	$\dfrac{\alpha_I I}{\sqrt[n]{K_I^n + (\alpha_I I)^n}}$	Bannister (1979); Laws and Bannister (1980)
V65_LIGHT	$\dfrac{\alpha_I I}{\sqrt{I_{opt}^2 + \alpha_I^2 I^2}} \dfrac{1}{[1 + (\beta \frac{I}{I_{opt}})^2]^{n/2}}$	Vollenweider (1965)
PE78_LIGHT	$\dfrac{I}{I_{opt}} \dfrac{2 + \alpha_I}{1 + \alpha_I \dfrac{I}{I_{opt}} + \left(\dfrac{I}{I_{opt}}\right)^2}$	Peeters and Eilers (1978)
WNS74_LIGHT	$1 - e^{-\frac{\alpha_I I}{\mu_{max}}}$	Webb et al. (1974)
PGH80_LIGHT	$(1 - e^{-\frac{\alpha_I I}{\mu_{max}}}) e^{-\frac{\beta I}{\mu_{max}}}$	Platt et al. (1980)
JP76_LIGHT	$\tanh(\dfrac{\alpha_I I}{\mu_{max}})$	Jassby and Platt (1976)

BWDC99_LIGHT	$\tanh[\dfrac{\alpha_I(I-I_o)}{\mu_{max}}]e^{\beta(I_{opt}-I)}$	Bissett et al. (1999)

Note: I_{opt} is the optimal light intensity at which the phytoplankton growth rate reaches its maximum value, I is the light intensity, I_o is the compensation light intensity, α_I is a light parameter related to the slope of the light function, μ_{max} is the maximum growth rate, and β is the parameter determining the photo inhibition rate. All of these equations are derived or proposed based on fits to observations, and the choice of function to use should be made with caution, particularly in a situation without any observational data for guidance.

In our biological module, the nutrient limiting function $\mu_i(N_i)$ is specified using the Michaelis-Menten form given as

$$\mu_i(N_j) = \begin{cases} \dfrac{N_j - N_{j\min}}{K_{js} + N_j - N_{j\min}} & N_j > N_{j\min} \\ 0 & N_j \le N_{j\min} \end{cases}, \quad j=1,2,3,\cdots,nn \qquad (11.24)$$

where N_j is the jth nutrient concentration, $N_{j\min}$ is the threshold of the jth nutrient concentration, and K_{js} is the half-saturation constant of the jth nutrient concentration. Note that the subscript "i" represents the ith phytoplankton species.

In the case when both ammonium and nitrate are considered, the phytoplankton usually prefer ammonium relative to nitrate. The limiting function of nitrate is usually inhibited by ammonium. A widely used expression for the nitrate limiting function with consideration of the inhibition factor due to ammonium is given here as

$$\mu_i(NO_3) = \begin{cases} \dfrac{NO_3 - (NO_3)_{\min}}{K_{NO_3} + NO_3 - (NO_3)_{\min}} \cdot \dfrac{K_{NH_4}}{K_{NH_4} + NH_4 - (NH_4)_{\min}} & \begin{cases} NO_3 > (NO_3)_{\min} \\ NH_4 > (NH_4)_{\min} \end{cases} \\ \\ \dfrac{NO_3 - (NO_3)_{\min}}{K_{NO_3} + NO_3 - (NO_3)_{\min}} & \begin{cases} NO_3 > (NO_3)_{\min} \\ NH_4 \le (NH_4)_{\min} \end{cases} \\ \\ 0 & NO_3 \le (NO_3)_{\min} \end{cases} \qquad (11.25)$$

The ammonium limiting function is given as

$$\mu_i(NH_4) = \dfrac{NH_4 - (NH_4)_{\min}}{K_{NH_4} + NH_4 - (NH_4)_{\min}} \qquad (11.26)$$

140

If only nitrate and ammonium concentrations are considered as nutrients, then the total nitrogen limitation is the sum of eqs. (11.25) [for $\mu_i(NO_3)$] and (11.26) [for $\mu_i(NH_4)$].

11.2.2.3. Zooplankton

Ocean and estuarine ecosystems generally feature multiple zooplankton species that are competitive in the lower trophic food web dynamics. Since it is very difficult to create a simple platform to include all heterotrophic processes involving zooplankton, particularly when various life stages of the different dominant zooplankton species are considered, we have chosen to include only the basic transformations involving zooplankton in the lower trophic food web, which are described next.
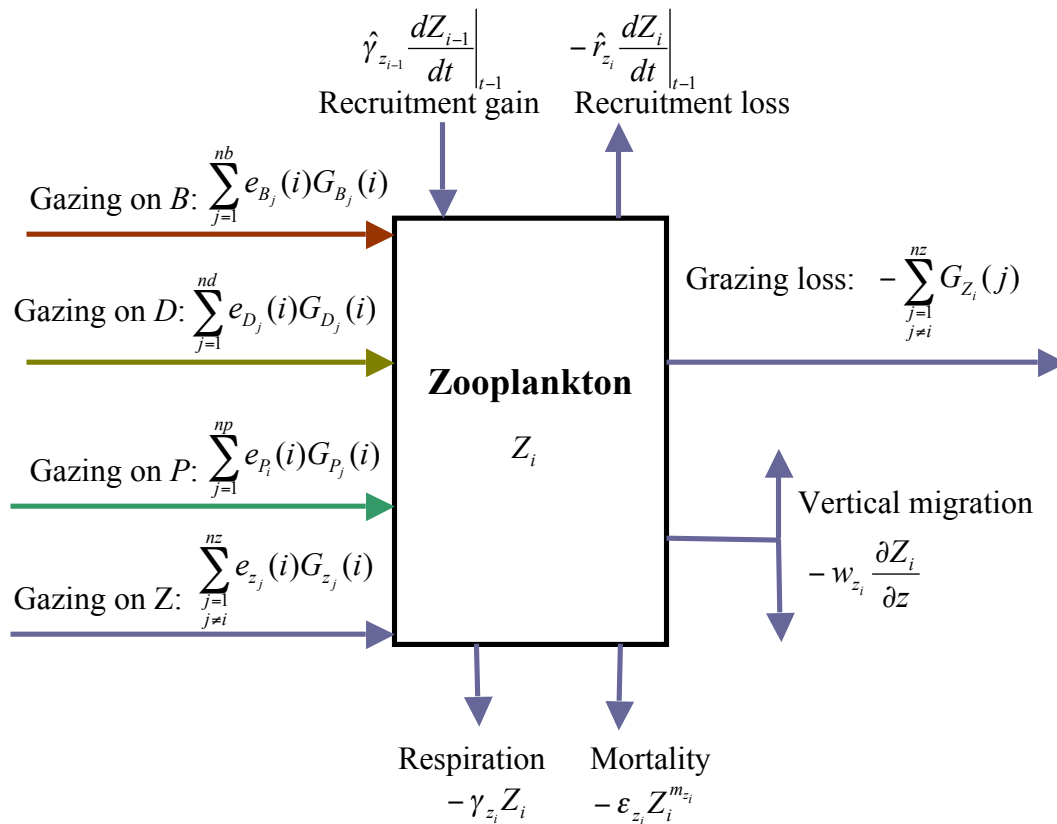
Fig. 11.4: Inflow/outflow chart of the ith zooplankton species Z_i.

The inflow/outflow chart for an individual zooplankton species and its interaction with other zooplankton is shown in Fig. 11.4. The governing equation for the ith zooplankton species Z_i is given as

$$\frac{\partial Z_i}{\partial t} = Grazing\_P + Grazing\_D + Grazing\_B + Grazing\_Z$$
$$+ Respiration\_Z_i + Mortality\_Z_i + Grazing\_Z_i + Migration\_Z_i \qquad (11.27)$$
$$+ Recruitment\ gain\_Z_i + Recruitment\ loss\_Z_i$$

where

$$Grazing\_P = \sum_{j=1}^{np} e_{P_j}(i)G_{P_j}(i) \qquad (11.28)$$

$$Grazing\_D = \sum_{j=1}^{nd} e_{D_j}(i)G_{D_j}(i) \qquad (11.29)$$

$$Grazing\_B = \sum_{j=1}^{nb} e_{B_j}(i)G_{B_j}(i) \qquad (11.30)$$

$$Grazing\_Z = \sum_{\substack{j=1 \\ j\neq i}}^{nz} e_{z_j}(i)G_{z_j}(i) \qquad (11.31)$$

$$Respiration\_Z_i = -\gamma_{Z_i} Z_i \qquad (11.32)$$

$$Mortality\_Z_i = -\varepsilon_{Z_i} Z_i^{m_{z_i}} \qquad (11.33)$$

$$Grazing\_Z_i = -\sum_{\substack{j=1 \\ j\neq i}}^{nz} e_{z_i}(j)G_{z_i}(j) \qquad (11.34)$$

$$Migration\_Z_i = -w_{z_i}\frac{\partial Z_i}{\partial z} \qquad (11.35)$$

$$Recruitment\ gain\_Z_i = \hat{\gamma}_{Z_{i-1}}\frac{dZ_{i-1}}{dt}\bigg|_{t-1} \qquad (11.36)$$

$$Recruitment\ loss\_Z_i = -\hat{\gamma}_{Z_i}\frac{dZ_i}{dt}\bigg|_{t-1} \qquad (11.37)$$

where $e_{P_j}(i)$, $e_{D_j}(i)$, $e_{B_j}(i)$, and $e_{z_j}(i)$ are the grazing efficiencies of Z_i on P_j, D_j, B_j, and other Z_j, γ_{Z_i} is the respiration rate of Z_i, ε_{Z_i} is the mortality rate of Z_i, w_{Z_i} is the vertical migration velocity (which can be specified to be a constant or vertical profile or time-dependent function related to light and food limitation), $\hat{\gamma}_{Z_i}$ is the recruitment success rate of Z_i, and m_{Z_i} is the power of Z_i for mortality. Detailed description of the units used for these parameters is given in Table 11.4

Table 11.4: Parameters used in the zooplankton equations

Parameter	Definition	Unit
$e_{P_j}(i)$	grazing efficiencies of Z_i on P_j	fraction
$e_{D_j}(i)$	grazing efficiencies of Z_i on D_j	fraction
$e_{B_j}(i)$	grazing efficiencies of Z_i on B_j	fraction
$e_{z_j}(i)$	grazing efficiencies of Z_i on Z_j	fraction
γ_{Z_i}	respiration rate of Z_i	s^{-1}
ε_{Z_i}	mortality rate of Z_i	s^{-1}
$\hat{\gamma}_{Z_i}$	recruitment success rate of Z_i	fraction
m_{Z_i}	power of Z_i for the mortality	dimensionless
$g_{\max}^{P_j}(i)$	maximum grazing rate of Z_i on P_j	s^{-1}
K_P	half-saturation constant	mmol C m^{-3}
$\sigma_{P_j}(i)$	preference coefficient of Z_i on P_j	(mmol C m^{-3})$^{-1}$
$\lambda_{P_j}(i)$	Ivelv constant for P_j, grazed by Z_i	(mmol C m^{-3})$^{-1}$
$\alpha_T^{Z_i}$	water temperature influence constant for Z_i grazing	(C°)$^{-1}$
$d_{\max}(i)$	maximum grazing rate of Z_i on D	s^{-1}
$\sigma_{D_j}(i)$	preference coefficient Z_i on D_j	(mmol C m^{-3})$^{-1}$
K_{D_j}	half saturation constant of D_j	mmol C m^{-3}
m_{D_j}	power of D_j used in the grazing function	dimensionless
w_R	random generation within a range from -1 to 1 with 50% probability at each migration time step	dimensionless
$w_{\max}$	maximum vertical migration speed	m s^{-1}
κ_{Z_i}	vertical migration constant	(mmol C m^{-3})$^{-1}$

$G_{P_j}(i), G_{D_j}(i), G_{B_j}(i)$ and $G_{Z_j}(i)$ are the grazing functions of Z_i on P_j, D_j, B_j, and Z_j ($j \neq i$). There are various types of functions used for grazing processes. For example, we can easily find multiple functions used for $G_{P_j}(i)$. The simplest one is the Lotka-Volterra function

$$G_{P_j}(i) = g_{P_j} P_j Z_i \tag{11.38}$$

where g_{P_j} is the grazing rate (Chen, 2003a). This function represents an unlimited grazing process without a saturation stage. It is used in some theoretical studies of the phytoplankton-zooplankton model, but not in real ocean applications. Attention should be paid when selecting the proper function in the case with multiple types of prey. For this reason, we include various choices of the grazing functions. These functions can be used for the single prey case by setting the prey number to 1. Detailed descriptions of the functions included in the BEM for P, D, B, and Z are given next.

a) $G_{P_j}(i)$

A review of the available literature shows that the existing grazing functions can be divided into three groups: 1) grazing on a single prey, 2) grazing on multiple prey, and 3) grazing on multiple prey with active switching. In most cases, functions used in group 1 are a special case of the functions in groups 2 and 3. Because many of the functions have similar behavior, we incorporate into BEM some of the functions from groups 2 and 3 to provide users with a suitable range of choices.

In general,

$$G_{P_j}(i) = g_{P_j}(i) Z_i \tag{11.39}$$

where $g_{P_j}(i)$ is the grazing function of Z_i for P_j. The various formulas of $g_{P_j}(i)$ are given in Table 11.5, and the new variables and their units are listed in Table 11.4.

Table 11.5: Grazing functions for zooplankton

Name	Function	References
IVLE1_G	$g_{P_j}(i) = g_{max}^{P_j}(i)(1 - e^{-\lambda_{P_i}(i)P_j})$	Ivlev (1955)

RECTI_G	$g_{P_j}(i) = \begin{cases} g_{\max}(i)\dfrac{\sigma_{P_j}(i)P_j}{K}, & \text{for } R \le K \\ g_{\max}(i)\dfrac{\sigma_{p_j}(i)P_j}{R}, & \text{for } R > K \end{cases}$, $\quad R = \sum_{j=1}^{np}\sigma_{P_j}(i)P_j$	Armstrong (1994)
CLI_G	$g_{P_j}(i) = g_{\max}(i)\sigma_{P_j}(i)P_j(1 - e^{-\sigma_{P_j}(i)P_j})$	Leonard et al. (1999)
IVLE2_G	$g_{p_j}(i) = g_{\max}(i)\left(1 - e^{-\lambda_P(i)R}\right)\dfrac{\sigma_{P_j}(i)P_j}{R}$, $\quad R = \sum_{j=1}^{np}\sigma_{P_j}(i)P_j$	Hofmann and Ambler (1988)
MM1_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{\sigma_{P_j}(i)P_j}{K_P + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j}$	Moloney and Field (1991)
MM2_G	$g_{P_j}(i) = g_{\max}(i)\left(\dfrac{R - P_c}{K_P + R - P_c}\right)\dfrac{\sigma_{P_j}(i)P_j}{R}$, $\quad R = \sum_{j=1}^{np}\sigma_{P_j}(i)P_j$	Evans (1988); Lancelot et al. (2002); Leising et al. (2003)
MM3_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{\sigma_{P_j}(i)P_j}{1 + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j}$	Verity (1991); Fasham et al. (1990); Strom and Loukos (1998); Tian et al. (2001)
SMM_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{\sigma_{P_i}(i)P_i^2}{K_P \sum_{j=1}^{np}\sigma_{P_j}(i)P_j + \sum_{j=1}^{np}\sigma_{P_j}(i)P_j^2}$	Fasham et al. (1990); Strom and Loukos (1998); Pitchford and Brindley (1999); Spitz et al. (2001)

GSF1_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{[\sigma_{P_j}(i)P_j]^{m_g}}{\displaystyle\sum_{j=1}^{np}[\sigma_{P_J}(i)P_j]^{m_g}}$	Tansky (1978); Matsuda et al. (1986)
GSF2_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{(\sigma_{P_j}(i)P_j)^{m_g}}{\left(\displaystyle\sum_{j=1}^{np}(\sigma_{P_j}(i)P_j)\right)^{m_g}}$	Vance (1978)
GSMM_G	$g_{P_j}(i) = g_{\max}(i)\dfrac{\{\sigma_{P_j}(i)[P_j-(P_j)_c\}^{m_g}}{1+\displaystyle\sum_{j=1}^{np}\{\sigma_{P_j}(i)[P_j-(P_j)_c]^{m_g}}$	Gismervik and Andersen (1997)

Note: $g_{\max}(i)$ is the maximum grazing rate of Z_i on P, K_P is the half-saturation constant, P_c is the threshold value below which grazing is zero, $\sigma_{P_j}(i)$ is the preference coefficient of Z_i on P_j, and $\lambda_{P_j}(i)$ is the Ivlev constant for P_j, grazed by Z_i. See Table 11.4 for a list of variables and their units.

In the code, we also consider the influence of water temperature on grazing. In this case,

$$g_{\max}(i) = \hat{g}_{\max}(i)e^{-\alpha_T^{Z_i}\left|T-T_{opt}(Z_i)\right|} \tag{11.40}$$

where $\alpha_T^{Z_i}$ is the water temperature influence constant for Z_i grazing.

b) $G_{D_j}(i)$

We include two methods to calculate $G_{D_j}(i)$ here. First, it is represented using the switching Michaelis Menten function given as

$$G_{D_j}(i) = d_{\max}(i)\frac{\{\sigma_{D_j}(i)[D_j-(D_j)_c]\}^{m_{D_j}}}{K_{D_j}+\displaystyle\sum_{j=1}^{nd}\{\sigma_{D_j}(i)[D_j-(D_j)_c]\}^{m_{D_j}}}D_j \tag{11.41}$$

where $d_{\max}(i)$ is the maximum grazing rate of Z_i on D_j, $\sigma_{D_j}(i)$ is the preference coefficient of Z_i on D_j, K_{D_j} is the half saturation constant of D_j, and m_{D_j} is the power of D_j. Secondly, we can add $G_{D_j}(i)$ into one component for the total grazing of Z_i (see the description given below in *e*).

c) $G_{B_j}(i)$

By replacing D with B, eq. (11.41) can be directly used to calculate $G_{B_j}(i)$. The grazing function of Z_i on B_j can be expressed as

$$G_{B_j}(i) = b_{\max}(i) \frac{\{\sigma_{B_j}(i)[B_j - (B_j)_c]\}^{m_{B_j}}}{K_{B_j} + \sum_{j=1}^{nb} \{\sigma_{B_j}(i)[B_j - (B_j)_c]\}^{m_{B_j}}} B_j \qquad (11.42)$$

where $b_{\max}(i)$ is the maximum grazing rate of Z_i on B_j, $\sigma_{B_j}(i)$ is the preference coefficient of Z_i on B_j, K_{B_j} is the half saturation constant of B_j, and m_{B_j} is the power of B_j. Alternatively, we can also add $G_{B_j}(i)$ into one component for the total grazing of Z_i (see the description given below in **e**).

d) $G_{Z_j}(i)$

The grazing of other zooplankton species Z_i on Z_j ($j \neq i$) can be estimated by replacing P_j by Z_j in the grazing functions listed in Table 11.5 or by replacing D_j or B_j in eq. (11.41) or (11.42) by Z_j. In general, to make the code clearer, we prefer to select the same form as those we choose for $G_{P_j}(i)$.

e) Combination of $G_{P_j}(i), G_{D_j}(i), G_{B_j}(i)$, and $G_{Z_j}(i)$

We have listed many choices for calculating $G_{P_j}(i), G_{D_j}(i), G_{B_j}(i)$, and $G_{Z_j}(i)$. If these grazing functions are computed using the different distinct formulas, it would involve a determination of many parameters. To make the code simpler, we also add an expression for the grazing of Z_i on phytoplankton, detritus, bacteria and other zooplankton species together using a generalized switching Michaelis-Menten function given as

$$R_j = \sum_{i=1}^{np} \{\sigma_{P_j}(i)[P_i - (P_j)_c]\}^{m_{P_j}} + \sum_{i=1}^{nd} \{\sigma_{D_j}(i)[D_i - (D_j)_c]\}^{m_{D_j}}$$

$$+ \sum_{i=1}^{np} \{\sigma_{B_j}(i)[B_i - (B_j)_c]\}^{m_{B_j}} + \sum_{\substack{i=1 \\ j \neq i}}^{nz} \{\sigma_{Z_j}(i)[Z_i - (Z_j)_c]\}^{m_{Z_j}} \qquad (11.43)$$

and

$$G_{P_j}(i) = \begin{cases} g_i(T)\dfrac{\{\sigma_{P_j}[P_j - (P_j)_c]\}^{m_{P_j}}}{1 + R_j} Z_i & \text{if } P_j > (P_j)_c \\ 0 & \text{if } P_j \le (P_j)_c \end{cases} \qquad (11.44)$$

$$G_{D_j}(i) = \begin{cases} g_i(T)\dfrac{\{\sigma_{D_j}[D_j - (D_j)_c]\}^{m_{D_j}}}{1 + R_j} Z_i & \text{if } D_j > (D_j)_c \\ 0 & \text{if } D_j \le (D_j)_c \end{cases} \qquad (11.45)$$

$$G_{B_j}(i) = \begin{cases} g_i(T)\dfrac{\{\sigma_{B_j}[B_j - (B_j)_c]\}^{m_{B_j}}}{1 + R_j} Z_i & \text{if } B_j > (B_j)_c \\ 0 & \text{if } B_j \le (B_j)_c \end{cases} \qquad (11.46)$$

$$G_{Z_j}(i) = \begin{cases} g_i(T)\dfrac{\{\sigma_{D_j}[Z_j - (Z_j)_c]\}^{m_{Z_j}}}{1 + R_j} Z_i & \text{if } Z_j > (Z_j)_c \\ 0 & \text{if } Z_j \le (Z_j)_c \end{cases} \qquad (11.47)$$

where

$$g_i(T) = \hat{g}_{\max}(i)e^{-\alpha_T^{Z_i}\left|T - T_{opt}(Z_i)\right|} \qquad (11.48)$$

It should be noted that the half saturation constant is scaled by the preference coefficient in eqs. (11.44)-(11.48). Therefore, the preference coefficient described here should be scaled by the half saturation constant.

f) *Grazing $\_Z_i$*

The loss of Z_i grazed by other zooplankton species should be parameterized using the same form shown above in ***d***.

g) Vertical migration velocity (w_{Z_i})

Many observations reveal diurnal vertical migration in many zooplankton species. The influence of vertical migration on the spatial distribution of zooplankton species has been widely studied in the past twenty years (Anderson and Nival, 1991). The driving mechanism is complicated, which is related to both environmental and biological factors such as light, food abundance, water temperature, salinity, gravity, available dissolved oxygen, turbulence, predation, species, age, sex, state of feeding, reproduction, energy conservation, and biological cycles, etc. (Forward, 1988; Ohman, 1990; Anderson and Nival, 1991). For example, some zooplankton species migrate beneath the euphotic layer to escape predation in daylight and then move back near the surface for feeding during

the night (Frank and Widder, 1997). This diurnal migration process can be parameterized by specifying w_{Z_i} using a diurnal period function related to the daily light variation. Since this is case dependent, we leave it as a "user specified" value in the code.

The vertical migration of zooplankton is also related to food abundance. For example, when zooplankton migrate upward to reach the maximum chlorophyll layer, they may not continue towards the surface layer where food is scare. If only food abundance is considered, the vertical migration can be estimated by

$$w_{Z_i} = w_R \cdot w_{\max}(1 - e^{-\kappa_{Z_i} TF})$$ (11.49)

where w_R is a random number in the range of -1 to 1 for 50% probability at each migration time step, $w_{\max}$ is the maximum migration velocity, and κ_{Z_i} is the constant vertical migration constant for the ith zooplankton Z_i and *TF* represents the total food abundance

11.2.2.4. Detritus

Detritus refers to fecal pellets, dead phytoplankton and zooplankton, unassimilated phytoplankton, other species of zooplankton and detritus due to zooplankton grazing as well as aggregation and disaggregration gains in the internal detritus cycling. The local change in detritus is caused by grazing loss by zooplankton on phytoplankton, other zooplankton, and bacteria, remineralization to nutrients, aggregation and disaggregation losses of the internal detritus cycling and vertical sinking. The inflow/outflow chart describing these processes is shown in Fig. 11.5.

The equation for the balance of these terms is written as

$$\frac{\partial D_i}{\partial t} = \text{P\_mortality} + \text{Z\_mortality} + \text{Feeding\_loss} + \text{Aggregation\_gain}$$
$$+ \text{Disaggregation\_gain} + \text{Grazing\_loss} + \text{Remineralization}$$
$$+ \text{Aggregation\_loss} + \text{Disaggregation\_loss} + \text{Vertical\_sinking}$$ (11.50)
$$+ \text{ADV\_D}_i + \text{HDIFF\_D}_i + \text{VDIFF\_D}_i$$

where
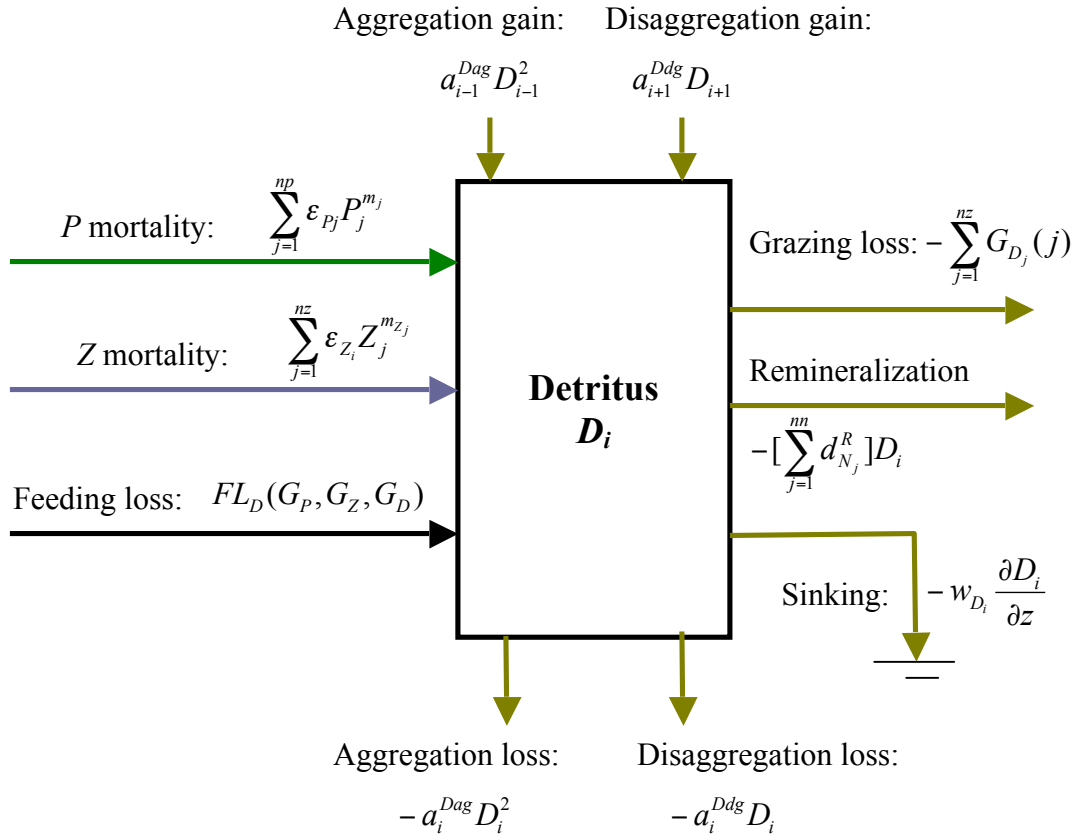
Fig. 11.5: Inflow/outflow chart for detritus D_i

$$P\_mortality = \sum_{j=1}^{np} \varepsilon_{P_j} P_j \tag{11.51}$$

$$Z\_mortality = \sum_{j=1}^{nZ} \varepsilon_{z_j} Z_j \tag{11.52}$$

$$Feeding\_loss = FL_D(G_P, G_z, G_D)$$

$$= \sum_{k=1}^{nz} [\sum_{j=1}^{np} \alpha_{P_j}^D(k) P_j + \sum_{\substack{j=1 \\ j \neq i}}^{nz} \alpha_{Z_j}^D(k) Z_j + \sum_{j=1}^{nd} \alpha_{B_j}^D(k) D_j] \tag{11.53}$$

$$Aggregation\_gain = a_{i-1}^{Dag} D_{i-1}^2 \tag{11.54}$$

$$Disaggregation\_gain = -a_{i+1}^{Ddg} D_{i+1}$$

$$\tag{11.55}$$

$$Grazing\_loss = -\sum_{j=1}^{nz} G_{D_i}(j) \tag{11.56}$$

$$Remineralization \ = -[\sum_{j=1}^{nn} d_{N_j}^R]D_i \tag{11.57}$$

$$Aggregation\_loss = -a_i^{Dag} D_i^2 \tag{11.58}$$

$$Disaggregation\_loss = a_i^{Ddg} D_i \tag{11.59}$$

$$Vertical\_sinking = -w_{D_i} \frac{\partial D_i}{\partial z} \tag{11.60}$$

where $\alpha_{P_j}^D(i)$, $\alpha_{Z_j}^D(i)$, and $\alpha_{D_j}^D(i)$ are coefficients of feeding loss to detritus by zooplankton grazing on P_j, Z_j, and D_j. w_{D_i} is the vertical sinking velocity of D_i. $d_{N_j}^R$ is the remineralization rate of D_i to nutrient N_j. The descriptions of parameters used in the detritus equations and their units are given in Table 11.6.

Table 11.6: Parameters used in the detritus equations

Parameter	Definition	Unit
$\alpha_{P_j}^D(i)$	coefficient of feeding loss to detritus by zooplankton grazing on phytoplankton P_j	fraction
$\alpha_{Z_j}^D(i)$	coefficient of feeding loss to detritus by zooplankton grazing on other zooplankton Z_j	fraction
$\alpha_{D_j}^D(i)$	coefficient of feeding loss to detritus by zooplankton grazing on detritus D_j.	fraction
w_{D_i}	vertical sinking velocity of D_i.	m s^{-1}
$d_{N_j}^R$	remineralization rate of D_i to nutrient N_j.	s^{-1}
$a_{D_i}^{Dag}$	aggregation coefficient of detritus D_i	s^{-1}
$a_{D_i}^{Ddg}$	disaggregation coefficient of detritus D_i	s^{-1}
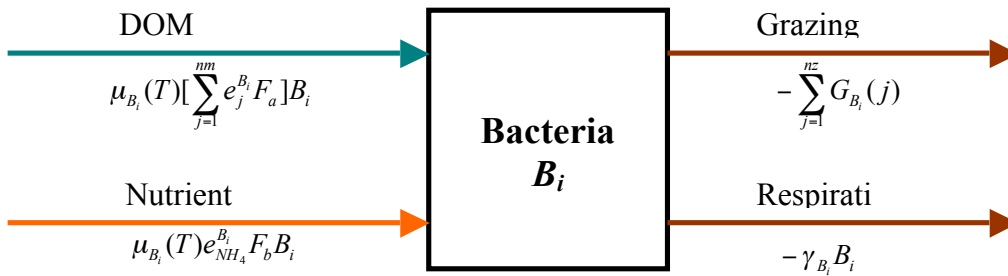
Fig. 11.6: The inflow/outflow chart for the bacteria equation for B_i

11.2.2.5 Bacteria

Bacteria contribute to the lower trophic level food web mainly through uptake of nutrients and decomposition of DOM, grazing by microzooplankton, and bacteria respiration. In GEM, no explicit form for the mortality of bacteria is included. Because the size of individual bacteria is very small, the usual loop in which dead bacteria deposit into the detritus pool and are subsequently remineralized into dissolved nutrients can be simplified by simply including it into the bacteria respiration term. For a nitrogen-limited system, the growth of bacteria described here is mainly controlled by DOM and NH_4 uptake. For a phosphate-limited system, the bacteria can also uptake phosphorus. To make the equations more general, we include these two cases in the code.

The bacteria equation can be expressed as

$$\frac{\partial B_i}{\partial t} = Uptake\_DOM + Uptake\_N + \text{Grazing loss}\_B_i + \text{Respiration}\_B_i$$
$$+ ADV\_B_i + HDIFF\_B_i + VDIFF\_B_i \tag{11.61}$$

a) Bacteria uptake

In a nitrogen-limited system,

$$Uptake\_DOM = \mu_{B_i}(T)[\sum_{j=1}^{nm} e_j^{B_i} F_a(DOM_j, NH_4)]B_i \tag{11.62}$$

where $e_j^{B_i}$ is the gross growth efficiency rate of the ith bacteria B_i, $\mu_{B_i}(T)$ is the growth rate of the ith bacteria B_i, and T is the water temperature. In the code, $\mu_{B_i}(T)$ is given as

$$\mu_{B_i}(T) = \mu_{\max}^{B_i} e^{-\alpha_T^{B_i}|T-T_{opt}|} \tag{11.63}$$

where $\mu_{\max}^{B_i}$ is the maximum growth rate of the ith bacteria B_i, T_{opt} is the optimal water temperature at which the maximum growth rate occurs, and $\alpha_T^{B_i}$ is the exponential decay rate of the temperature limiting factor.

$$F_a(DOM_j, NH_4) =$$

$$
\begin{cases}
\dfrac{\sigma_j^{DOM}[DOM_j - (DOM_j)_c]}{1 + \displaystyle\sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(DON, NH_4)} & \text{if } DOM_j > (DOM_j)_c \\
\\
0 & \text{if } DOM_j \le (DOM_j)_c
\end{cases}
\tag{11.64}
$$

and

$$\Psi(DON, NH_4) =$$

$$
\begin{cases}
Min \begin{cases} \sigma_{B_i}^{NH_4}[NH_4 - (NH_4)_c], \\ \\ \delta_{B_i} \displaystyle\sum_{j=1}^{nm} \sigma_j^{B_i}[DON_j - (DON_j)_c] \end{cases} & \text{if } \begin{array}{l} NH_4 > (NH_4)_c \\ DON_j > (DON_j)_c \end{array} \\
\\
0 & \text{if } \begin{array}{l} NH_4 \le (NH_4)_c \\ DON_j \le (DON_j)_c \end{array}
\end{cases}
\tag{11.65}
$$

where σ_j^{DOM} is the preference coefficient for DOM_j, the subscript "c" means the critical value for the minimum uptake, $\sigma_{B_i}^{NH_4}$ is the preference coefficient of B_i for NH_4, $\sigma_j^{B_i}$ is the preference coefficient for the jth DON_j, and δ_{B_i} is the uptake ratio of NH_4^+ to DON.

δ_{B_i} can be given as either a constant value or calculated by

$$
\delta_{B_i} =
\begin{cases}
\dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} - 1 & \dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} > 1 \\
\\
0 & \dfrac{\overline{e}^{B_i}(N:C)_{B_i}}{e_{B_i}^{NH_4}(N:C)_{DOM}} \le 1
\end{cases}
\tag{11.66}
$$

where

$$\overline{e}^{B_i} = \frac{1}{nm} \sum_{j=1}^{nm} e_j^{B_i} \tag{11.67}$$

In a nitrogen-limiting system, $Uptake\_N$ is expressed by $Uptake\_NH_4$. The nutrient uptake is considered only in the situation when $\displaystyle\sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] > 0$. In this case,

$$Uptake\_NH_4 = \mu_{B_i}(T)e_{B_i}^{NH_4} F_b(DOM, NH_4)B_i \tag{11.68}$$

where

$$F_b(DOM, NH_4) = \frac{\Psi(DON, NH_4)}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(DON, NH_4)} \quad (11.69)$$

In the case when *Uptake _ DOM* is equal to zero,

$$Uptake \_ NH_4 \equiv 0 \quad (11.70)$$

In a phosphorus-limiting system (like rivers and lakes), eq. (11.69) remains unchanged except that $F_a(DOM_j, NH_4)$ is replaced by $F_a(DOM_j, PO_4)$. $F_a(DOM_j, PO_4)$ is given as

$$F_a(DOM_j, PO_4) =$$

$$\begin{cases} \dfrac{\sigma_j^{DOM}[DOM_j - (DOM_j)_c]}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(PO_4)} & \text{if } DOM_j > (DOM_j)_c \\ 0 & \text{if } DOM_j \le (DOM_j)_c \end{cases} \quad (11.71)$$

and

$$\Psi(PO_4) = \begin{cases} \sigma_{B_i}^{PO_4}[PO_4 - (PO_4)_c] & \text{if } PO_4 > (PO_4)_c \\ 0 & \text{if } PO_4 \le (PO_4)_c \end{cases} \quad (11.72)$$

where $\sigma_{B_i}^{PO_4}$ is the preference coefficient of B_i for PO_4.

Uptake _ N is expressed by *Uptake _ PO_4*. The nutrient uptake is considered only in the situation when $\sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] > 0$. In this case,

$$Uptake \_ PO_4 = \mu_{B_i}(T) e_{B_i}^{PO_4} F_b(DOM, PO_4) B_i \quad (11.73)$$

where

$$F_b(DOM, PO_4) = \frac{\Psi(PO_4)}{1 + \sum_{j=1}^{nm} \sigma_j^{DOM}[DOM_j - (DOM_j)_c] + \Psi(PO_4)} \quad (11.74)$$

b) Grazing loss

The grazing loss of B_i caused by microzooplankton is presented in the form of

$$Grazing\ loss\_B_i = -\sum_{j=1}^{nz} G_{B_i}(j) \quad (11.75)$$

where $G_{B_i}(j)$ is the grazing loss of B_i by the jth zooplankton species.

c) Respiration loss

The respiration loss of the ith bacteria is given as

$$Respiration\ loss\_B_i = -\gamma_{B_i} B_i \qquad (11.76)$$

where γ_{B_i} is respiration rate of B_i.

Table 11.7: Parameters used in the bacteria equations

Parameter	Definition	Unit
$e_j^{B_i}$	gross growth efficiency rate of the ith bacteria B_i for the jth DOM_j	fraction
$\mu_{max}^{B_i}$	maximum growth rate of the ith bacteria B_i	s^{-1}
T_{opt}	optimal water temperature at which the maximum growth rate occurs	°C
$\alpha_T^{B_i}$	exponential decay rate of the temperature limiting factor	(°C)$^{-1}$
σ_j^{DOM}	preference coefficient for DOM_j	(mmol C m^{-3})$^{-1}$
$\sigma_{B_i}^{NH_4}$	preference coefficient for the ith B_i	(mmol N m^{-3})$^{-1}$
$\sigma_j^{B_i}$	preference coefficient for the jth DOM_j	(mmol N m^{-3})$^{-1}$
γ_{B_i}	B_i respiration rate	s^{-1}

11.2.2.6. DOM

The variation in the DOM concentration in the ocean is a complex process. In our GEM, the local change in DOM concentration is controlled by phytoplankton passive and active exudations, detritus dissolution, feeding loss from bacteria, phytoplankton, zooplankton, and detritus, and ageing gain via bacteria uptake and ageing loss. An inflow/outflow chart for DOM_i is shown in Fig. 11.5 and the equation is given as

$$\frac{\partial DOM_i}{\partial t} = P\_passive\_exudation + P\_active\_exudation + D\_dissolution$$
$$+ Feeding\_loss + Ageing\_gain - Ageing\_loss - B\_uptake$$
$$+ ADV\_DOM_i + HDIFF\_DOM_i + VDIFF\_DOM_i$$

$$(11.77)$$

where ageing gain and loss are internal processes between multiple DOM_i. In the majority of realistic ocean ecosystem studies, DOM is normally treated as a single variable in which the internal ageing process is not considered. Each term in eq. (11.77) is described below:

$$P\_passive\_exudation = \sum_{j=1}^{np} d_{P_j} P_j \tag{11.78}$$

$$P\_active\_exudation = \sum_{j=1}^{np} d_j^{DOM_i} \mu_{P_j} P_j \tag{11.79}$$

$$D\_dissolution = \sum_{j=1}^{nd} d_j^D D_j \tag{11.80}$$

$$Ageing\_gain = a_{i-1}^{DOM_i} DOM_{i-1} \tag{11.81}$$

$$Ageing\_loss = -a_i^{DOM_i} DOM_i \tag{11.82}$$

$$B\_uptake = -\mu_B(T) \sum_{j=1}^{nb} F_a B_j \tag{11.83}$$
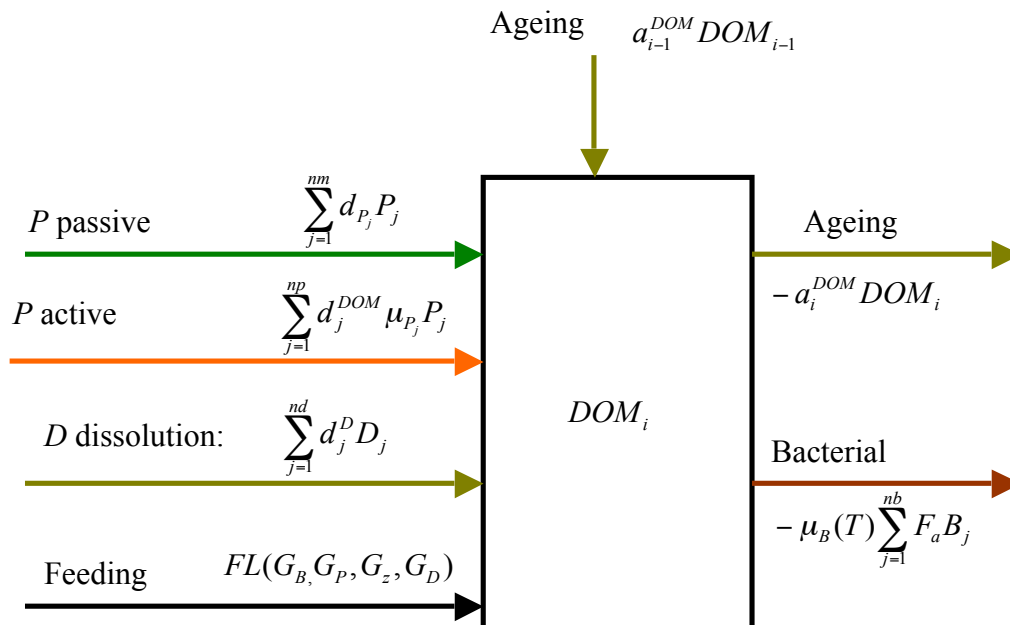
Fig. 11.7: Inflow/outflow chart for DOM_i equation.

where $a_i^{DOM_i}$ is the ageing coefficient of DOM_i for the case when DOM is defined based on its bioavailability. The definitions of other parameters are given in the phytoplankton and detritus equations. The feeding loss is the sum of the unassimilated portions of phytoplankton, zooplankton, detritus and bacteria defined as

$$FL(G_P, G_z, G_B, G_D) = \sum_{k=1}^{nz} [\sum_{j=1}^{np} \alpha_{P_j}(k)P_j + \sum_{\substack{j=1 \\ j \neq i}}^{nz} \alpha_{Z_j}(k)Z_j$$

$$+ \sum_{j=1}^{nb} \alpha_{B_j}(k)B_j + \sum_{j=1}^{nd} \alpha_{D_j}(k)D_j]$$

(11.84)

where $\alpha_{P_j}(i)$, $\alpha_{Z_j}(i)$, $\alpha_{B_j}(i)$ and $\alpha_{D_j}(i)$ are coefficients of grazing loss by Z_i on phytoplankton, other zooplankton, bacteria and detritus.

The parameters used in this equation are listed in Table. 11.11.

Table 11.8: Parameters used in the DOM equations

Parameter	Definition	Unit
$a_i^{DOM_i}$	ageing coefficient of DOM_i	s^{-1}
$\alpha_{P_j}(i)$	coefficient of grazing loss by Z_i on phytoplankton	fraction
$\alpha_{Z_j}(i)$	coefficient of grazing loss by Z_i on other zooplankton	fraction
$\alpha_{B_j}(i)$	coefficient of grazing loss by Z_i on bacteria	fraction
$\alpha_{D_j}(i)$	coefficient of grazing loss by Z_i on detritus	fraction

To help users understand the GEM code, we have summarized all variables and parameters into Table. 11.9. Their implementation in the module is the same as that shown in this table.

Table 11.9: Definitions and units of variables and parameters used in the GEM

Name	Definition	Unit

B_i	concentration of the ith bacteria	mmol C m^{-3}
D_i	concentration of the ith detritus	mmol C m^{-3}
DOM_i	concentration of the ith DOM	mmol C m^{-3}
N_i	concentration of the ith nutrient	mmol C m^{-3}
P_i	concentration of the ith phytoplankton	mmol C m^{-3}
Z_i	concentration of the ith zooplankton	mmol C m^{-3}
$a_i^{DOM_i}$	ageing coefficient of the ith DOM	s^{-1}
α	weight coefficient (0,1) for phytoplankton growth	dimensionless
α_I	light parameter related to the slope of the light function (see in Table 11.3)	dimensionless in MM-, LB-, V65-, PE78-LIGHT; mmol C(mg Chla)$^{-1}$(Wm^{-2})$^{-1}$s^{-1}, in WNS74-, PGH80-, JP76-, BWDC99-LIGHT
a_i^{Dag}	aggregation coefficient of detritus D_i	s^{-1}
a_i^{Ddg}	disaggregation coefficient of detritus D_i	s^{-1}
$\alpha_{B_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on bacteria B_j	fraction
$\alpha_{D_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on detritus D_j	fraction
$\alpha_{P_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on P_j	fraction
$\alpha_{Z_j}(k)$	coefficient of grazing loss of grazing loss by Z_k on Z_j	fraction
$\alpha_{B_j}^D(k)$	unassimilated coefficient for detritus by zooplankton grazing on detritus B_j.	fraction
$\alpha_{D_j}^D(i)$	unassimilated coefficient for detritus by zooplankton grazing on detritus D_j.	fraction
$\alpha_{P_j}^D(i)$	unassimilated coefficient for detritus by zooplankton grazing on phytoplankton P_j	fraction
$\alpha_{Z_j}^D(i)$	unassimilated coefficient for detritus by zooplankton grazing on other zooplankton Z_j	fraction
α_T	exponential decay rate of $\mu_i(T)$ for P	(°C)$^{-1}$
$\alpha_T^{B_i}$	exponential decay rate of the temperature	(°C)$^{-1}$

	limiting factor for B_i	
$\alpha_T^{Z_i}$	exponential decay rate of the temperature limiting factor for Z_i	$(°C)^{-1}$
β	Photo inhibition coefficient in PGH80 and BWDC99_LIGHT functions	mmol C(mg Chla)$^{-1}$(Wm^{-2})$^{-1}$s^{-1}
d_j^D	dissolution rate of detritus D_j	s^{-1}
d_i^{DOM}	DOM active exudation of P_i	fraction
$d_{max}(i)$	maximum grazing rate of Z_i on D	s^{-1}
$d_{N_j}^R$	remineralization rate of D_i by nutrient N_j	s^{-1}
d_{P_i}	DOM passive exudation rate of P_i	s^{-1}
$e_j^{B_i}$	gross growth efficiency rate of the ith bacteria B_i for the jth DOM_j	fraction
$e_{B_i}^{NH_4}$	growth efficiency of B_i on NH$_4^+$	fraction
$e_{B_j}^{PO_4}$	growth efficiency of B_i on PO$_4^{3-}$	fraction
$e_{B_j}(i)$	grazing efficiencies of Z_i on B_j	fraction
$e_{D_j}(i)$	grazing efficiencies of Z_i on D_j	fraction
$e_{P_j}(i)$	grazing efficiencies of Z_i on P_j	fraction
$e_{z_j}(i)$	grazing efficiencies of Z_i on Z_j	fraction
ε_{P_i}	mortality rate of P_i	s^{-1}
ε_{Z_i}	mortality rate of Z_i	s^{-1}
$g_{max}^{P_j}(i)$	maximum grazing rate of Z_i on P_j	s^{-1}
$\Gamma_{N_i:C}^{B_j}$	ratio of the ith nutrient N_i unit to carbon for the jth bacteria B_j.	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{D_j}$	ratio of the ith nutrient N_i unit to carbon for the jth detritus D_j.	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{DOM_j}$	ratio of the ith nutrient N_i unit to carbon for the jth DOM_j.	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{P_j}$,	ratio of the ith nutrient N_i unit to carbon for the jth phytoplankton P_j.	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{Z_j}$	ratio of the ith nutrient N_i unit to carbon for the jth zooplankton Z_j.	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{Z_kB_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on B_j	mmol N$_i$:mmol C
$\Gamma_{N_i:C}^{Z_kD_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on D_j	mmol N$_i$:mmol C

$\Gamma_{N_i:C}^{Z_k P_j}$,	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on P_j	mmol N_i:mmol C
$\Gamma_{N_i:C}^{Z_k Z_j}$	ratio of the ith nutrient N_i unit to carbon for active respiration and unassimilated material from Z_k grazing on Z_j	mmol N_i:mmol C
γ_{B_i}	respiration rate of the ith bacteria B_i	s^{-1}
γ_{P_i}	respiration rate of the ith phytoplankton P_i;	s^{-1}
γ_{Z_j}	passive respiration rate of the ith zooplankton Z_i;	s^{-1}
$\hat{\gamma}_{Z_i}$	recruitment success rate of Z_i	Fraction
$I_{\max}$	maximum light intensity at the surface	Wm^{-2}
I_o	light intensity at the sea surface	Wm^{-2}
I_{opt}	optimal light for phytoplankton	Wm^{-2}
k	light attenuation coefficient	m^{-1}
K	saturation for Z grazing in RECTI_G	mmol C m^{-3}
K_{D_j}	half saturation constant of D_j	mmol C m^{-3}
K_{B_j}	half saturation constant of B_j	mmol C m^{-3}
K_I	half-saturation of the light-P growth function	Wm^{-2}
K_P	half-saturation for grazing in MM1_G, MM2_G, SMM_G,	mmol C m^{-3}
K_{js}	half-saturation of nutrient N_j	mmol N_j m^{-3}
κ_{Z_i}	vertical migration constant	(mmol C m^{-3})$^{-1}$
$\lambda_{P_j}(i)$	Ivlev constant for P_j grazed by Z_i	(mmol C m^{-3})$^{-1}$
m_I	power of P_i in the mortality term	dimensionless
m_{D_j}	power of D_j used in the grazing function	dimensionless
m_{Z_i}	Power of Z_i for the mortality.	dimensionless
$\mu_{\max}^{B_i}$	maximum growth rate of the ith bacteria B_i	s^{-1}
$\mu_{\max}^{P_i}$	maximum growth rate of the ith bacteria P_i	mmol C (mg Chla)$^{-1}$ s^{-1}
$\mu_{\max}$	parameter in WNS74_LIGHT, PGH80_LIGHT, BWDC90_LIGHT	mmol C (mg Chla)$^{-1}$ s^{-1}
μ_{P_j}	gross growth rate of phytoplankton P_i	mmol C (mg Chla)$^{-1}$ s^{-1}
μ_I	gross growth rate of phytoplankton P_i	mmol C(mg Chl)$^{-1}$s^{-1}
N	power of light in LB_LIGHT, V65_LIGHT	dimensionless

	functions	
$N_{j\min}$	threshold of nutrient Nj	mmol N_j m^{-3}
$(N:C)_{B_i}$	N:C ratio in bacteria B_i	mmol N/mmol C
$(N:C)_{DOM}$	N:C ratio in bacteria DOM_i	mmol N/mmol C
σ_j^{DOM}	preference coefficient for DOM_j	(mmol C m^{-3})$^{-1}$
$\sigma_j^{B_i}$	preference coefficient for the ith B_i	(mmol N m^{-3})$^{-1}$
$\sigma_{B_i}^{NH_4}$	preference coefficient of B_i for NH_4	(mmol N m^{-3})$^{-1}$
$\sigma_{B_i}^{PO_4}$	preference coefficient of B_i for PO_4	(mmol P m^{-3})$^{-1}$
$\sigma_{B_j}(i)$	preference coefficient Z_i on B_j	dimensionless
$\sigma_{D_j}(i)$	preference coefficient Z_i on D_j	dimensionless
$\sigma_{P_j}(i)$	preference coefficient of Z_i on P_j	dimensionless
$\theta_{i(chl:N_i)}$	ratio of chlorophyll a to carbon in P_i	mg Chl / mmol C
$T_{opt}^{B_i}$	optimal water temperature at which the maximum growth rate occurs for bacteria B_i	°C
T_{opt}	optimal temperature for phytoplankton	°C
$T_{opt}(Z_i)$	optimal temperature for zooplankton Z_i	°C
TF	total food abundance for vertical migration	(mmol C m^{-3})$^{-1}$
w_{D_i}	vertical sinking velocity of D_i.	m s^{-1}
$w_{\max}$	maximum vertical migration speed	m s^{-1}
w_{P_i}	sinking velocity of P_I	m s^{-1}
w_R	random generation within a range from -1 to 1 with 50% probability at each migration time step	dimensionless
w_{z_i}	vertical migration velocity of Z_i	m s^{-1}

11.3. Pre-built Biological Models

   FVCOM includes several types of companion finite-volume biological and water quality model modules within FVCOM. These models can be run either online (together with the FVCOM physical model) or offline (driven by FVCOM output). To provide users with information about these biological and water quality models, brief descriptions of their formulations are given in this chapter.

11.3.1. The Nutrient-Phytoplankton-Zooplankton (NPZ) Model

The NPZ model is a simple model developed by Franks et al. (1986). This model was originally coupled into ECOM-si for the study of phytoplankton dynamics in the tidal mixing front on Georges Bank (Franks and Chen, 1996 and 2001), and coupled into FVCOM as an option for a simple biological module in 2002. The schematic of the food web loop in the NPZ model is shown in Fig. 11.8 and the governing equations of this model are given in the form of

$$\frac{dP}{dt} - \frac{\partial}{\partial z}(K_h \frac{\partial P}{\partial z}) = \frac{V_m N}{k_s + N} f(I_o)P - ZR_m(1 - e^{-\lambda P}) - \varepsilon P + F_P \tag{11.85}$$

$$\frac{dZ}{dt} - \frac{\partial}{\partial z}(K_h \frac{\partial Z}{\partial z}) = \gamma ZR_m(1 - e^{-\lambda P}) - gZ + +F_z \tag{11.86}$$

$$\frac{dN}{dt} - \frac{\partial}{\partial z}(K_h \frac{\partial N}{\partial z}) = -\frac{V_m N}{k_s + N} f(I_o)P + (1 - \gamma)ZR_m(1 - e^{-\lambda P}) + \varepsilon P + gZ + F_N \tag{11.87}$$

where V_m is the maximum phytoplankton growth rate, k_s is the half-saturation constant for phytoplankton growth, R_m is the maximum grazing rate on phytoplankton by zooplankton, λ is the grazing efficiency on phytoplankton by zooplankton, γ is the
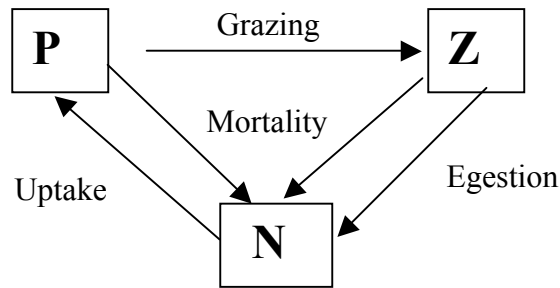
Fig. 11.8: Schematic of the food web loop of the NPZ model

fraction of ingested phytoplankton unassimilated by zooplankton, g is the zooplankton death rate, ε is the phytoplankton death rate, K_h is the vertical diffusion coefficient for N, P, and Z, and F_P, F_Z, and F_N are the horizontal diffusion terms for P, Z, and N.

This NPZ model presents a simple food web in which dissolved nutrients are taken up by phytoplankton following Michaelis-Menten kinetics, and phytoplankton are grazed by zooplankton through an Ivlev functional response.

Some modifications have been made on the original formulation of the NPZ model used by Franks et al. (1986). First, instead of using a constant mortality rate, we include the Steele and Henderson (1992) and Kawamiya et al. (1995) mortality formulas in which ε is proportional to the biomass of P in the form of

$$\varepsilon = \delta P \tag{11.88}$$

where δ is the new (non-constant) mortality rate that was assumed to be constant before. Numerical experiments have revealed that this mortality rate formula is robust enough to model a conservative biological system in cases without extra sources and sinks. Second, a spatially-dependent diffuse attenuation coefficient of irradiance is used, which allows the model to capture the spatial variation of the photosynthesis process due to the spatial difference in light penetration.

The boundary conditions of N, P, and Z are specified by the users based on the scientific problems to be addressed. For a case with no flux at the surface and bottom, the boundary conditions for N, P, and Z are given as

$$\frac{\partial P}{\partial z} = \frac{\partial Z}{\partial z} = \frac{\partial N}{\partial z} = 0, \quad \text{at } z = \zeta(t,x,y) \text{ and } -H(x,y) \tag{11.89}$$

The initial distributions of the biological state variables N, P, and Z must be specified according to either field data or some simple theory. In many cases, they are specified by using the steady state solution of the NPZ model.

The biological parameters used to drive the NPZ model must also be specified. Since these parameters vary over a wide range with time and space, these parameters must be selected from observational data or theoretical considerations. Some examples can be seen in our previous modeling efforts made on Georges Bank (Franks and Chen, 1996 and 2001), Jiaozhou Bay (Chen et al., 1999), and the Louisiana and Texas shelf (Chen et al., 1997).

Equations (11.85)-(11.89) have been converted to the σ-coordinate system and are solved by the finite-volume method using the same basic numerical method used to solve the FVCOM temperature equation.

11.3.2. The Phosphorus-Controlled Lower Trophic Level Food Web Model

A phosphorus-controlled ecosystem model was developed by Chen et al. (2002). The schematic of this model is shown in Fig. 11.9, which was built based on the observed features of the lower trophic level food web in Lake Michigan (Scavia and Fahnenstiel, 1987). The governing equations are given as:

$$\frac{dP_L}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial P_L}{\partial z}) = LP(\text{uptake}) - LP(\text{mortality}) - LZLP(\text{grazing}) - LP(\text{sinking}) \qquad (11.90)$$

$$\frac{dP_S}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial P_S}{\partial z}) = SP(\text{uptake}) - SP(\text{mortality}) - SZSP(\text{grazing}) \qquad (11.91)$$

$$\frac{dZ_L}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial Z_L}{\partial z}) = \varepsilon^{Z_L}\sigma_P LZLP(\text{grazing}) + \varepsilon^{Z_{LS}} LZSZ(\text{grazing}) - LZ(\text{mortaity}) \qquad (11.92)$$

$$\frac{dZ_S}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial Z_S}{\partial z}) = \varepsilon^{Z_S} SZSP(\text{grazing}) - LZSZ(\text{grazing}) + \varepsilon^{B} SZB(\text{grazing}) - SZ(\text{mortality}) \qquad (11.93)$$

$$\frac{dB}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial B}{\partial z}) = DB(\text{decomposition}) + BP(\text{uptake}) - SZB(\text{grazing}) - B(\text{mortality}) \qquad (11.94)$$

$$\frac{dD_S}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial D_P}{\partial z}) = \sigma_S LZLP(\text{grazing}) + \sigma_S LP(\text{mortality}) - DS(\text{remineralization}) \qquad (11.95)$$

$$\frac{dD_P}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial D_P}{\partial z}) = (1-\varepsilon^{z_L})\sigma_p LZLP(\text{grazing}) + (1-\varepsilon^{Z_S})SZSP(\text{grazing})$$
$$+ (1-\varepsilon^{B})SZB(\text{grazing}) + (1-\varepsilon^{Z_{LS}})LZSZ(\text{grazing})$$
$$- DB(\text{decomposition}) - DP(\text{sinking}) - DP(\text{remineralization}) \qquad (11.96)$$
$$+ \sigma_P LP(\text{mortality}) + SP(\text{mortality}) + LZ(\text{mortality})$$
$$+ SZ(\text{mortality}) + B(\text{mortality})$$

$$\frac{dP}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial P}{\partial z}) = -\sigma_p LP(\text{uptake}) - SP(\text{uptake}) - BP(\text{uptake})$$
$$+ DP(\text{remineralization}) + PQ \qquad (11.97)$$

$$\frac{dS_i}{dt} - \frac{\partial}{\partial z}(A_h \frac{\partial S_i}{\partial z}) = -\sigma_S LP(\text{uptake}) + DS(\text{remineralization}) + SQ \qquad (11.98)$$

where P_L, P_S, Z_L, Z_S, B, P, and S_i are the large-size phytoplankton, small-size phytoplankton, large-size zooplankton, small-size zooplankton, bacteria, phosphorus, and silicon, respectively. D_P and D_S are the phosphate- and silica-related components of detritus. A_h is the thermal diffusion coefficient that is calculated using the Mellor and Yamada level 2.5 turbulent closure scheme incorporated in the physical model.
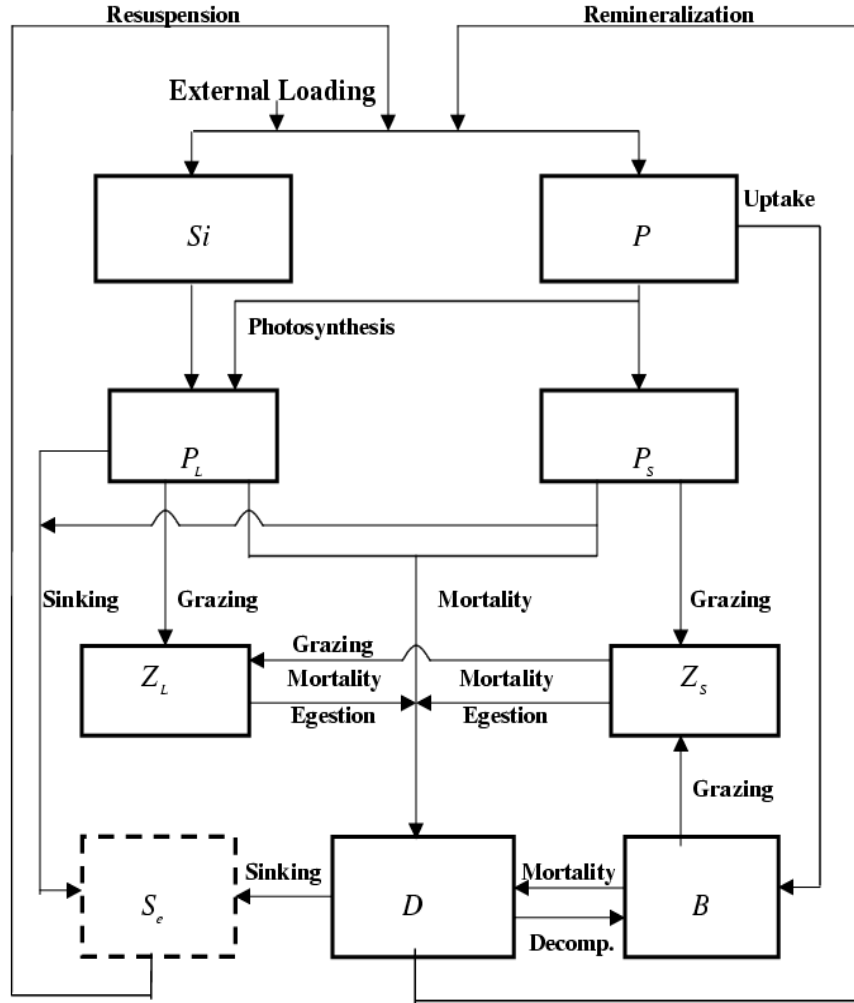
Fig. 11.9: Schematic of the NPZDB model.

$\dfrac{d}{dt} = \dfrac{\partial}{\partial t} + u\dfrac{\partial}{\partial x} + v\dfrac{\partial}{\partial y} + w\dfrac{\partial}{\partial z}$ is the derivative operator; x, y and z are the eastward, northward, and vertical axes in the Cartesian coordinate frame, and u, v, and w are the x, y, and z components of the velocity. The definition of the parameters ε^{Z_L}, $\varepsilon^{Z_{LS}}$, ε^{Z_S},

ε^B, σ_S, and σ_P are given in Table 11.10. *PQ* and *SQ* are the phosphorus and silicon fluxes from suspended sediments.

The mathematical formulas for *LP*(uptake), *LZLP*(grazing), *SP*(uptake), *SZSP*(grazing), *LZSZ*(grazing), *SZB*(grazing), *DB*(decomposition), *BP*(uptake), *DP*(remineralization), *DS*(remineralization), *LP*(sinking), *DP*(sinking), *LP*(mortality), *SP*(mortality), *LZ*(mortality), *SZ*(mortality), and *B*(mortality) are given as

$$LP\,(\text{uptake}) = \min\,(V_{\max}^{P_L}\,\frac{P}{k_{P_L}+P},V_{\max}^{S}\,\frac{S}{k_S+S})f(I)P_L \tag{11.99}$$

$$LZLP\,(\text{grazing}) = G_{\max}^{Z_L}\,(1-e^{-k^{P_L}P_L})Z_L \tag{11.100}$$

$$SP\,(\text{uptake}) = V_{\max}^{P_s}\,\frac{P}{k_{P_L}+P}\,f(I)P_s \tag{11.101}$$

$$SZSP\,(\text{grazing}) = G_{\max}^{Z_s}\,(1-e^{-k^{P_s}P_s})Z_s \tag{11.102}$$

$$LZSZ(\text{grazing}) = G_{\max}^{Z_{LS}}\,(1-e^{-k^{Z_{LS}}Z_s})Z_L \tag{11.103}$$

$$SZB\,(\text{grazing}) = G_{\max}^{B}\,(1-e^{-k^{B}B})Z_s \tag{11.104}$$

$$BP\,(\text{uptake}) = V_{\max}^{B}\,\frac{P}{k_B+P}\,B \tag{11.105}$$

$$DB(\text{decomposition}) = V_{\max}^{DOP}\,\frac{\mu_D D}{k_{DOP}+\mu_D D} \tag{11.106}$$

$$SZB\,(\text{grazing}) = G_{\max}^{B}\,(1-e^{-k^{B}B})Z_s \tag{11.107}$$

$$DP(\text{remineralization}) = e_P D_P \tag{11.108}$$

$$DS(\text{remineralation}) = e_s D_s \tag{11.109}$$

$$LP\,(\text{sinking}) = w_{P_L}\,\frac{\partial P_L}{\partial z} \tag{11.110}$$

$$LP\,(\text{mortality}) = \alpha^{P_L} P_L^2 \tag{11.111}$$

$$SP\,(\text{mortality}) = \alpha^{P_s} P_s^2 \tag{11.112}$$

$$LZ\,(\text{mortality}) = \alpha^{Z_L} Z_L^2 \tag{11.113}$$

$$SZ\,(\text{mortality}) = \alpha^{Z_s} Z_s^2 \tag{11.114}$$

$$B\,(\text{mortality}) = \alpha^{B} B^2 \tag{11.115}$$

where the definition for each parameter used in the above equations is given in Table 11.10. σ_P and σ_S are the phosphorus and silica fractions of large phytoplankton (diatoms) contained in the total amount of unassimilated zooplankton grazing, respectively, with $\sigma_P + \sigma_S = 1$. The value of σ_P is made according to the observed ratios of carbon to phosphorus in general plants and diatoms and then σ_S is given directly by $1 - \sigma_P$.

The dependence of the phytoplankton growth rate on incident irradiance intensity $f(I)$ is given as

$$f(I) = e^{-k_o z} \tag{11.116}$$

where k_o is the diffuse attenuation coefficient. Eq. (11.116) is normalized using the surface incident irradiance intensity. In general, the response of phytoplankton to light intensity varies according to different species. For many phytoplankton species, photosynthesis reaches its saturation level at a certain level of light intensity and then is inhibited as light intensity continues to become stronger. The linear assumption of $\ln f(I)$, used widely in previous lower trophic level food web models (Totterdell, 1993), does not consider saturation and inhibition of photosynthesis via light. It is found to be a good approximation in a vertically well-mixed region (Franks and Chen, 1996; Chen et al., 1997; Chen et al., 1999), but users should be aware of its limitation to reproduce the vertical profile of primary production which normally exhibits a maximum value at a subsurface depth. The Steele and Henderson (1992) mortality formula is used for phytoplankton and zooplankton. This empirical formula assumes that the organism mortality was proportional to its biomass.

Bacteria assimilation of dissolved organic phosphorus (DOP) is also assumed to follow the Michaelis-Menten function, in which DOP is proportional to the total detritus. This assumption is similar to having the bacteria graze detritus directly with a half-saturation constant of k_{DOP}/μ_D. A large portion of bacterial ingestion, which may be excreted into the particular organic pool, is taken into account in our numerical experiments by assuming a larger mortality rate for bacteria (Cotner and Wetzel, 1992).

Default values and their ranges of the biological parameters are listed in Table 11.10. These values were obtained either from field measurements made in Lake Michigan or from the literature. Since the biological parameters can vary in a wide range with time and space, a sensitivity analysis in parameter space should be conducted as part of the parameter setup.

In this model, phosphorus and silicon are the two limiting nutrients that control the primary production. Nitrogen was excluded since it was always found in sufficient concentration in the Great Lakes. This phosphorus-controlled food web model, in view of the food web system, had an advantage of avoiding the complex ammonia dynamics (Fasham et al., 1990).

Table 11.10: Biological Model Parameters

Parameter	Definition	Value used	Ranges	Sources
$V_{max}^{P_L}$	Maximum growth rate for P_L	1.6 d^{-1}	0.8-6 d^{-1}	Bieman and Dolan (1981); Scavia et al. (1988)
$V_{max}^{P_S}$	Maximum growth rate for P_S	1.2 d^{-1}	0.8-2 d^{-1}	Bieman &Dolan (1981); Scavia et al. (1988)
V_{max}^{S}	Maximum Si uptake rate by P_L	1.2d^{-1}	0.8-6 d^{-1}	Various sources
V_{max}^{B}	Maximum P uptake rate by B	0.05d^{-1}	?	
V_{max}^{DOP}	Maximum DOP uptake rate by B	5d^{-1}	23-144d^{-1}	Bentzen et al. (1992)
k_{P_L}	Half-saturation constant for the P uptake by P_L	0.2 μmol P/l	0.07-0.4 μmol P/l	Tilman et al. (1982); Bieman& Dollan (1981)
k_{P_S}	Half-saturation constant for the P uptake by P_S	0.05μmol P/l	0.015-?μmol P/l	Bieman &Dollan (1981)
k_{S}	Half-saturation constant for the Si uptake by P_L	5.0 μmol Si/l	3.5-3.57 μmol Si/l	Bieman&Dollan (1981); Jorgensen et al. (1991)
k_{B}	Half-saturation constant for the P uptake by B	0.2μmol P/l	0.02-0.2 μmol P/l	Cotner &Wetzel (1992)
k_{DOP}	Half-saturation constant for the DOP uptake by B	0.1μmol P/l	0.005-0.02 μmol P/l	Bentzen et al. (1992)
$G_{max}^{Z_L}$	Maximum P_L grazing rate by Z_L	0.4d^{-1}	0.2-	Jorgensen et al. (1991); Scavia et al.

			0.86d^{-1}	(1988)
$G_{\max}^{Z_S}$	Maximum P_S grazing rate by Z_S	0.2d^{-1}	0.1d^{-1}	Bieman&Dollan (1981)
$G_{\max}^{B}$	Maximum B grazing rate by Z_S	3.5d^{-1}	3.5d^{-1}	Hamilton&Preslan (1970)
$G_{\max}^{Z_{LS}}$	Maximum Z_S grazing rate by Z_L	0.4d^{-1}	?	
k^{Z_L}	Ivlev constant for Z_L grazing	0.06 l/μmol	0.001-1 l/μmol	Jorgensen et al.(1991); Scavia et al.(1988)
k^{Z_S}	Ivlev constant for P_s grazing by Z_s	0.02 l/μmol	0.011l/μmol	Bierman&Dollan(1981)
k^{B}	Ivlev constant for the B grazing by Z_s	0.03 l/μmol	0.022 l/μmol	Hamilton& Preslan (1970)
$k^{Z_{LS}}$	Ivlev constant for the Z_s grazing by Z_L	0.07	?	
ε^{Z_L}	Assimilation efficiency of Z_L	0.35	0.15-0.5	Jorgensen et al. (1991)
ε^{Z_S}	Assimilation efficiency of Z_s	0.3	?	
ε^{B}	Assimilation efficiency of B grazing by Z_s	0.3	?	
$\varepsilon^{Z_{LS}}$	Assimilation efficiency of the Z_s by Z_L	0.6	?	
α^{Z_L}	Mortality rate of Z_L	0.02d^{-1}	0.01-0.05 d^{-1}	Jorgensen et al. (1991); Bierman& Dollan (1981)
α^{Z_S}	Mortality rate of Z_s	0.03d^{-1}	0.1d^{-1}	Bierman&Dollan(1981)
□	Mortality rate of B	0.5d^{-1}	0.5-5.9 d^{-1}	Jorgensen et al. (1991)
□	Photosynthetic attenuation coefficient	0.08 m^{-1}	0.12-0.17 m^{-1}	Scavia et al. (1986)
□	Proportionality of DOP from the detrital P	0.02	0.1-0.58	Valiela (1995)
□	Sinking velocity of P_L	0.6 m d^{-1}	0.5-9 m d^{-1}	Jorgensen et al. (1991); Scavia et al. (1988)
□	Sinking velocity of P_s	0.3 m d^{-1}	0.01-3 m d^{-1}	Fahnenstiel & Scavia (1987)

δ_D	Sinking velocity of D	0.6 m d^{-1}	0.5-1 m d^{-1}	Jorgensen et al.(1991)
e_P	Remineralization rate of detrital P	0.15d^{-1}	0.05 d^{-1}	Fasham *et al.* (1990)
e_S	Remineralization rate of detrital Si	0.03d^{-1}	?	
α	Temperature dependence coefficient	0.069	0.069	Parsons et al. (1984)
$\lambda_{C:Chl_i}$	Ratio of carbon (C) to chlorophyll	35	23-79	Parsons et al. (1984)
$\lambda_{C:P}$	Ratio of C to P	80	?	Parsons et al. (1984)

11.3.3. The Multi-Species NPZD Model

A lower trophic level food web model was developed for a study of the spring bloom dynamics in the Gulf of Maine/Georges Bank region (Ji, 2003). This is a 9-compartment NPZD model including 3 nutrients (nitrate, ammonia and silicate), 2 phytoplankton (large- and small-size groups), 2 zooplankton (large- and small-size groups), 1 detrital organic nitrogen component and 1 detrital

Figure 11.10: Schematic of the flow chart of the multi-species NPZD model.

organic silicon component. The schematic of the model is given in Figure 11.10. As an extension of our previous NPZ model, this NPZD model was designed to resolve the seasonal pattern of the phytoplankton, while understanding the limitations of this model

in capturing the zooplankton dynamics that are linked directly to population dynamics. Unlike the carbon-based GEM, this NPZD model is based on nitrogen.

The governing equations of this 9-compartment model are given as

$$\frac{dP_S}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial P_S}{\partial z}\right) = F3 + F4 - F5 - F6 \tag{11.117}$$

$$\frac{dP_L}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial P_L}{\partial z}\right) = F1 + F2 - F11 - F14 \tag{11.118}$$

$$\frac{dZ_S}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial Z_S}{\partial z}\right) = F5a + F9 - F7 - F8 \tag{11.119}$$

$$\frac{dZ_L}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial Z_L}{\partial z}\right) = F8a + F14b - F12 \tag{11.120}$$

$$\frac{dNO_3}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial NO_3}{\partial z}\right) = -F1 - F3 \tag{11.121}$$

$$\frac{dNH_4}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial NH_4}{\partial z}\right) = F10 - F2 - F4 \tag{11.122}$$

$$\frac{dD_N}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial D_N}{\partial z}\right) = F5b + F6 + F7 + F8b$$
$$+ F10 + F11 + F12 + F14a - F9 - F10 \tag{11.123}$$

$$\frac{dSi}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial S_i}{\partial z}\right) = F17 - F13 \tag{11.124}$$

$$\frac{dD_{Si}}{dt} - \frac{\partial}{\partial z}\left(A_h \frac{\partial D_{Si}}{\partial z}\right) = F15 + F16 - F17 \tag{11.125}$$

where P_S is the small-phytoplankton biomass (μmol N l^{-1}), P_L the large-phytoplankton biomass (μmol N l^{-1}), Z_S the small-zooplankton biomass (μmol N l^{-1}), Z_L the large-zooplankton biomass (μmol N l^{-1}), NO_3 the nitrate concentration (μmol N l^{-1}), NH_4 the ammonium concentration (μmol N l^{-1}), D_N the particulate organic nitrogen concentration (μmol N l^{-1}), Si the silicate concentration (μmolSi l^{-1}), and Dsi the particulate organic silica concentration (μmolSi l^{-1}). $F1$ to $F17$ are the flux terms among the food web components defined as

$F1$: Uptake of nitrate by large phytoplankton (μmol N l^{-1}day^{-1})

$F2$: Uptake of ammonia by large phytoplankton (μmol N l^{-1}day^{-1})

$F3$: Uptake of nitrate by small phytoplankton (μmol N l^{-1}day^{-1})

$F4$: Uptake of ammonia by small phytoplankton (μmol N l^{-1}day^{-1});

*F*5: Small zooplankton grazing on small phytoplankton (μmol N l^{-1}day^{-1});

*F*5a: Assimilated part of *F*5 (μmol N l^{-1}day^{-1});

*F*5b: Un-assimilated part of *F*5 (μmol N l^{-1}day^{-1});

*F*6: Mortality of small phytoplankton (μmol N l^{-1}day^{-1});

*F*7: Mortality of small zooplankton (μmol N l^{-1}day^{-1});

*F*8: Large zooplankton grazing on small zooplankton (μmol N l^{-1}day^{-1});

*F*8a: *Assimilated part of F*8 (μmol N l^{-1}day^{-1});

*F*8b: Un-assimilated part of *F*8 (μmol N l^{-1}day^{-1});

*F*9: Small zooplankton grazing on detritus nitrogen (μmol N l^{-1}day^{-1});

*F*10: Remineralization of particulate organic nitrogen (μmol N l^{-1}day^{-1});

*F*11: Mortality of large phytoplankton (in term of N) (μmol N l^{-1}day^{-1});

*F*12: Mortality of large zooplankton (μmol N l^{-1}day^{-1});

*F*13: Uptake of silicate by large phytoplankton (μmol Si l^{-1}day^{-1});

*F*14: Large zooplankton grazing on large phytoplankton (μmol N l^{-1}day^{-1});

*F*14a: Assimilated part of *F*14 (μmol N l^{-1}day^{-1});

*F*14b: Un-assimilated part of *F*14 (μmol N l^{-1}day^{-1});

*F*15: Mortality of large phytoplankton (in term of Si) (μmol Si l^{-1}day^{-1});

*F*16: Silica rejected from large zooplankton (μmol Si l^{-1}day^{-1});

*F*17: Dissolution of particulate organic silica (μmol Si l^{-1}day^{-1}).

The mathematical formulas for *F*1, *F*3, …, *F*17 are given in Ji (2003). Many of them are very similar to those listed in eqs. (11.99)-(11.115).

In this model, the small-size phytoplankton group represents nano- and pico-sized phytoplankton, usually flagellates. The growth of small phytoplankton is limited by nitrogen and light. The large phytoplankton size group is explicitly modeled as diatoms that are limited by nitrogen, silicon and light. For application to Georges Bank, the zooplankton are dominated in abundance and biomass by the copepods *Calanus finmarchicus*, *Pseudocalanus sp.*, *Paracalanus parvus*, *Centropages typicus*, *Centropages hamatus*, and *Olithona similes*. At any given time of the year, these six species collectively make up over 80% of the total zooplankton abundance (Davis, 1987; Bryant et al., 1997). During the spring bloom period, the large zooplankton group in the model represents the dominant species of *Calanus* and *Pseudocalanus*. The small

zooplankton group refers to micro-zooplankton with size much smaller than the large zooplankton species.

Users should understand that no higher trophic level regulation on zooplankton is included in this model. Therefore, the large and small zooplankton in the model function like flux balancing terms to maintain the stability of the lower trophic level food web.

11.4 The Water Quality Models
11.4.1 FVCOM-WQM

The water quality model (WQM) described here is a modified version of the EPA Water Quality Analysis Simulation Program (WASP) (Ambrose et al., 1995). The benthic flux from sediment resuspension via sedimentation processes is added at the bottom to include the impact of the nutrient fluxes from the benthic layer to the water column (Zheng et al., 2004). The schematic of the WQM is shown in Fig. 11.11. This is a typical eutrophication model consisting of eight water quality state variables: (1) ammonia (NH_3), (2) nitrate and nitrite (NO_2 and NO_3), (3) inorganic phosphorus (OPO_4), (4) organic nitrogen (ON), (5) organic phosphorus (OP), (6) phytoplankton (PHYT), (7) carbonaceous biochemical oxygen demand (CBOD), and (8) dissolved oxygen (DO). The benthic and sediment resuspension processes are incorporated into the water model by adding a benthic layer and a sediment pool on the bed of the estuary or coastal water body. This biological/chemical model incorporates the basic transformation processes including photosynthesis, uptake, respiration, nitrification, benthic flux, sediment resuspension, and external loading. The modified water quality model was developed based on the characteristics of biological and chemical processes in Georgia estuaries, with the assistance of biologists at the University of Georgia (UGA), Skidaway Institute of Oceanography, and the Environmental Protection Agency (EPA) in Athens. The key references include Ambrose et al. (1995), Di Toro et al. (1971), Thomann et al. (1974), Di Toro and Matystik (1980), Di Toro and Connolly (1980), Thomann and Fitzpatrick (1982), and Di Toro and Fitzpatrick (1993).

The governing equations of the WQM are described in detail in Zheng et al. (2004). The model was converted to the σ-coordinate and then coupled to the ECOM-si physical model by a scientific team led by C. Chen at UGA. The unstructured-grid finite-volume

code version of WQM was written by L. Qi and C. Chen at SMAST and parallelized by L. Qi and G. Cowles. This program is coupled to FVCOM and can be run in either online or offline mode.
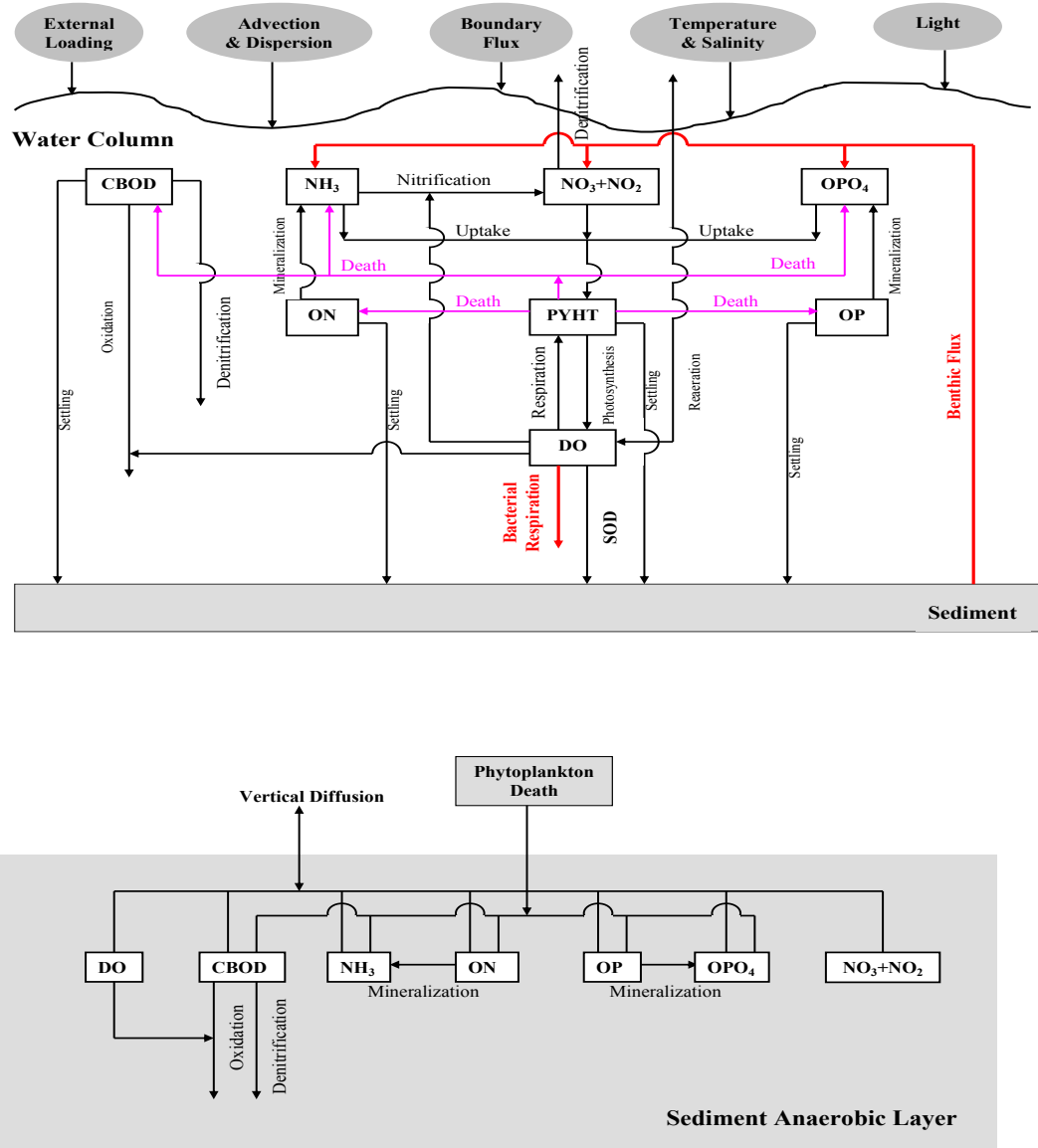
Fig. 11.11. Schematic of the standard EPA water quality model with inclusion of the benthic fluxes.

The WQM was set up in FVCOM as an independent parallelized module. This module was tested by running it for the Satilla River DO, nutrients and phytoplankton simulation. This module was released in version 2.5 of FVCOM and updated in FVCOM version 3.1.6.

11.4.2 UG-RCA

   UG-RCA is the unstructured-grid finite-volume version of RCA modified by the SMAST FVCOM development team under contract to the Massachusetts Water Resource Authority (MWRA) (Chen et al. 2010). RCA is the structured-grid version of the Row-Column Advanced water quality model developed by HydroQual (HydroQual, 2000, 2004). It consists of 26 water quality state variables and 23 sediment variables (Fig.
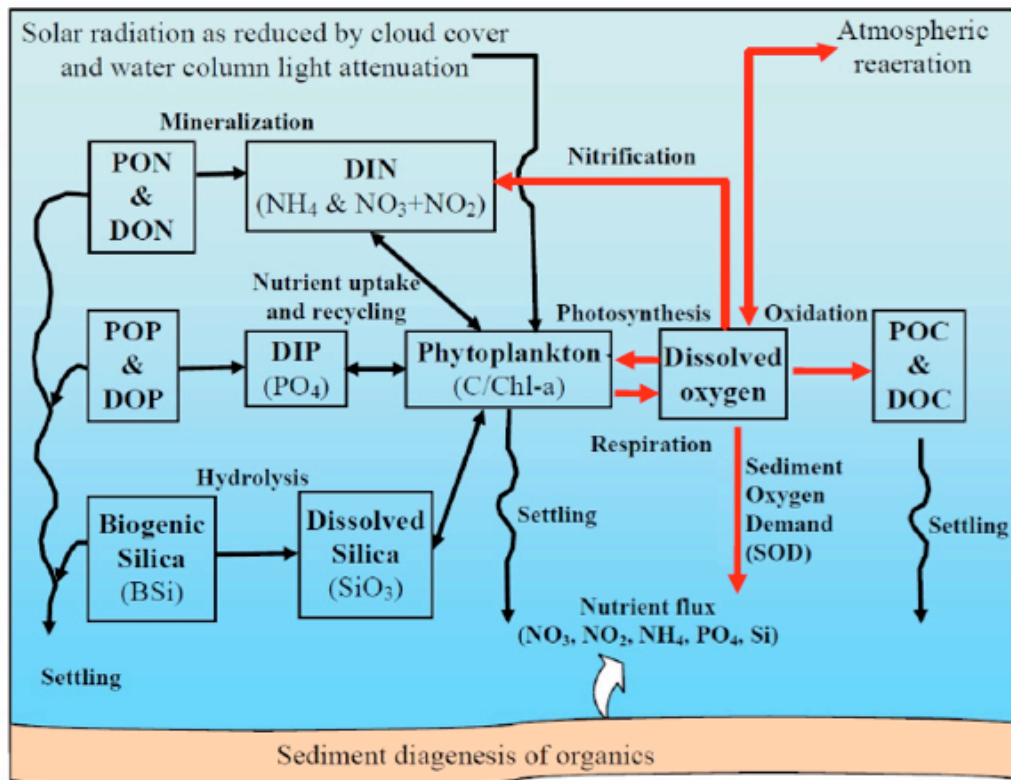
Fig. 11.12: Schematic of the UG-RCA model implemented into FVCOM (Chen et al., 2008).

11.12), including three phytoplankton assemblages (spring, summer and fall groups), four nutrients (ammonia, nitrate/nitrite, phosphate and dissolved silica), four organic phosphorus forms, four organic nitrogen pools, six organic carbon pools (four labile and refractory dissolved and particulate forms plus the reactive and exudates components), biogenic silica, dissolved and aqueous oxygen, and total active metal. In the model, nutrient and carbon loadings from point sources, non-point sources (e.g., ground water), rivers and atmosphere are the major anthropogenic perturbations to the system. DO is

computed by the surface flux through reaeration, bottom flux through sediment oxygen demand (SOD) and biogeochemical processes of oxidation of organic matter, nitrification, and photosynthesis-respiration by phytoplankton. In addition to the photosynthesis-respiration process, the growth of phytoplankton is also controlled by uptake of dissolved inorganic nutrients (including ammonium NH_4^+, nitrate NO_3^- and nitrite NO_2^-, phosphate PO_4^{3-} and dissolved silica (e.g. $Si(OH)_4$). The loss of phytoplankton is transformed into organic matter through "grazing", mortality, and exudation. The nutrient regeneration is produced by either remineralization of organic matter into inorganic nutrients in the water column or digenesis after settling down into sediment and re-enter water column through the sediment-water interface.

We developed UG-RCA with the objective of establishing an unstructured-grid Massachusetts Bay Eutrophication Model (BEM) system. A comprehensive suite of biological, chemical and sedimentological variables is routinely observed within the framework of the MWRA-sponsored monitoring program. The data from this monitoring program were used to determine the initial conditions, boundary conditions, anthropogenic forcing and model validation for the UG-RCA simulation. The model has successfully reproduced observed magnitudes and seasonal cycles of DO, phytoplankton, primary production, nutrients and nutrient flux at the sediment-water interface for 1995-2010 (Chen et al., 2010; Zhao et al., 2011; Xue et al., 2012). We converted RCA to the unstructured-grid finite-volume version (UG-RCA) using the same algorithms as FVCOM. We chose to base UG-RCA on RCA-v3, the newest version of RCA, because it is better supported, publically accessible, and has more options for light attenuation, phytoplankton growth functions, and reaeration than in earlier versions. UG-RCA is written in the MPI parallelized framework, so it can run in both online and offline modes. Under agreement with HydroQual, UG-RCA will serve as a publically accessible community water quality module under the FVCOM framework.

UG-RCA has been used for the MWRA annual water quality assessment since 2006. Driven by the Massachusetts Bay FVCOM output, this model has successfully reproduced the temporal and spatial variability of DO in this region. The computational domain of the Massachusetts Bay UG-RCA is shown in Fig. 11.13 and an example of the

DO simulation results for 1995-2010 is summarized in Fig. 11.14 with comparison to observations.
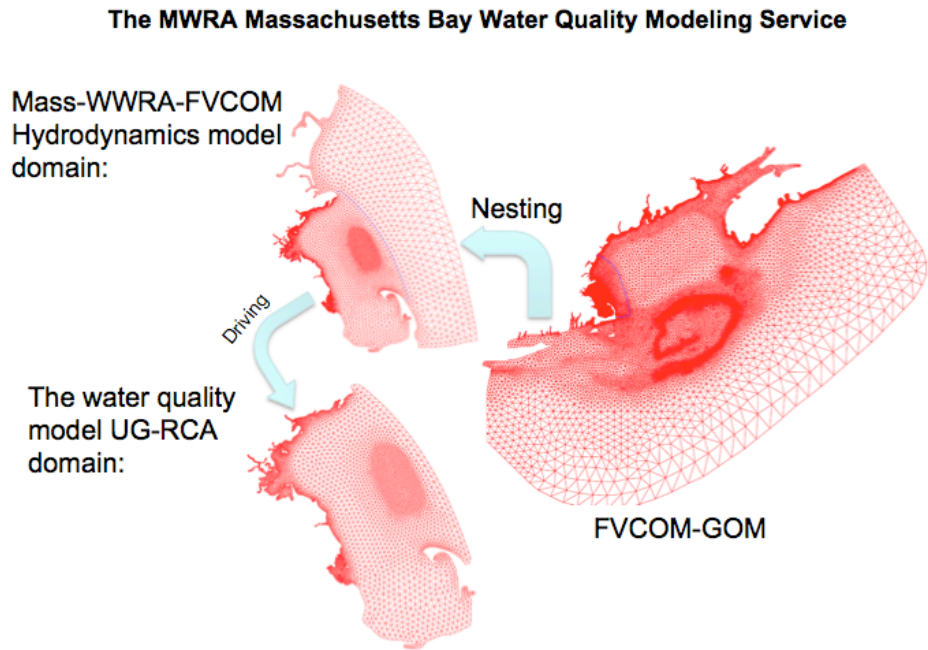
Fig.11.13: The Massachusetts Bay eutrophication model system driven by a multi-domain nested FVCOM.
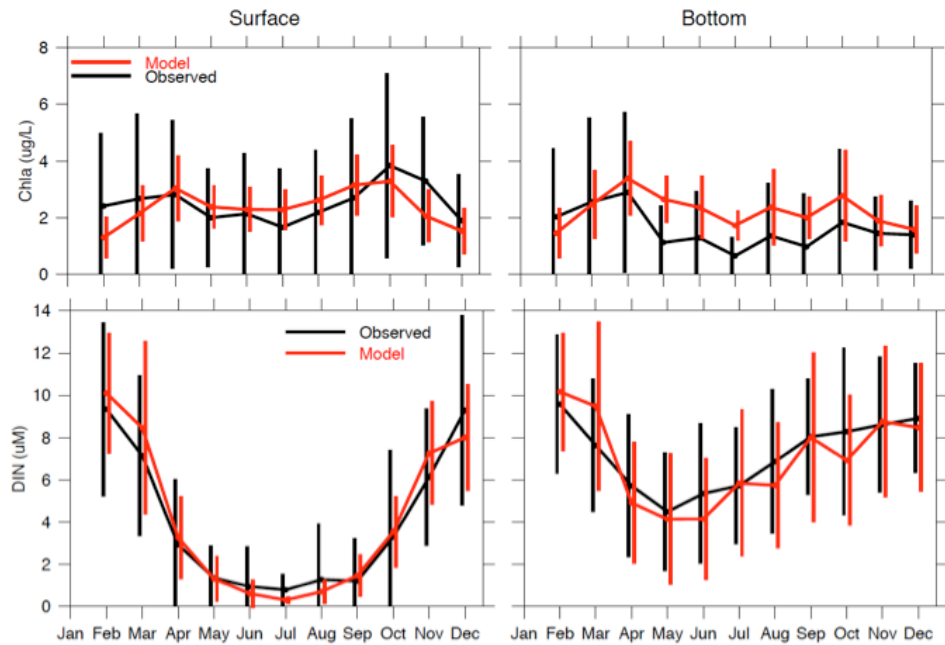
Fig. 11.14: Comparison of UG-RCA-computed and observed monthly averaged DO concentrations in the Massachusetts Bay for 1995-2010.

11.4.3 UG-CE-QUAL-ICM

UG-CE-QUAL-ICM is an unstructured-grid, finite-volume version of the Army Corps of Engineers water quality model CE-QUAL-ICM. CE-QUAL-ICM was originally developed by Cerco and Cole (1993) to study the hypoxia problem in Chesapeake Bay and is continuously being upgraded (Cerco et al., 2003; Cerco and Noel, 2004; Cerco et al., 2006). The model consists of 27 state variables, including multiple forms of phytoplankton, zooplankton, carbon, nitrogen, phosphorus, silica, DO, a pathogen, and two toxicants. Similar to UG-RCA, it also accounts for DO and nutrient fluxes between the sediment and water column. We converted CE-QUAL-ICM to an unstructured grid version and named it UG-CE-QUAL-ICM. This code was then tested, modified and validated by Drs. Taeyun Kim and Tarang Khangaonkar at the Pacific Northwest National Laboratory (they called it "FVCOM-CE-QUAL-ICM"). The code is available as an offline model in the FVCOM software package and described in detail in Kim et al. (2010).

Chapter 12: The Tracer-Tracking Model

The tracer-tracking equation incorporated in FVCOM is the same as the water temperature equation except with the addition of a source for the tracer. One could easily add this piece of the code by adding a source term in the temperature equation. To reduce any confusion that might arise using the same equation for temperature and tracer, we include a separate module for tracer tracking with the form

$$\frac{\partial DC}{\partial t} + \frac{\partial DuC}{\partial x} + \frac{\partial DvC}{\partial y} + \frac{\partial \omega C}{\partial \sigma} - \frac{1}{D}\frac{\partial}{\partial \sigma}(K_h \frac{\partial C}{\partial \sigma}) - DF_c = DC_o(x,y,\sigma,t) \qquad (12.1)$$

where C is the concentration of the tracer, D is the total water depth , $u, v,$ and ω are the x, y and σ components of the water velocity, K_h is the vertical thermal diffusion coefficient, F_c is the horizontal diffusion term, and C_o is the tracer concentration injected from a point source given as

$$C_o(x,y,\sigma,t) = \begin{cases} 1 & t_s \le t \le t_e; \sigma_k \le \sigma \le \sigma_{k+n}; x = \{x_i\}; y = \{y_i\}; i = 1, N \\ 0 & \text{otherwise} \end{cases} \qquad (12.2)$$

where t_s and t_e are the start and end time of the tracer injection, σ_k and σ_{k+n} refer to the upper and lower-bound σ-levels in which the tracer is injected, n can be an integral from 0 to KB-1 (KB is the total number of the σ-levels specified in the model), i is the node ID, and N is the total number of nodes where the tracer is injected. K_h is the vertical diffusion coefficient that is calculated using the turbulent closure scheme selected in FVCOM and the horizontal diffusivity in F_c is calculated using the Smagorinsky eddy parameterization method (Smagorinsky, 1963) in FVCOM.

This tracer-tracking module was originally developed when we applied FVCOM to simulate the evolution of a patch of dye that was released near the tidal mixing front on Georges Bank and tracked by R. Houghton (LDGO) as part of the 1999 North Atlantic/Georges Bank GLOBEC field program. The code was recently upgraded to run in parallel, so that it can be run either simultaneously (online mode) with the hydrodynamic part of FVCOM on single or multi-processor computers or later in offline mode.

In general, after the tracer is injected into the ocean in a field experiment, time series measurements of the tracer concentration are made as the ship attempts to follow the tracer patch in time and space. We have found that in order to make accurate comparisons of the observed tracer concentration data with the model simulation, the model tracer field needs to be sampled

in the same way as the in situ tracer field is sampled in the field, i.e., to sample the model at the same time and location as the ship sampled the ocean. To facilitate this, we have developed a model-sampling module that allows us to output the tracer concentration at selected times and locations defined by the ship tracking survey. This module can be operated in either online or offline mode. At present, this module works for single processor computers, and will be upgraded to parallel operation in the near future.

Chapter 13: The 3-D Lagrangian Particle Tracking Module

The Lagrangian particle tracking module consists of solving a nonlinear system of ordinary differential equations (ODE) as follows

$$\frac{d\vec{x}}{dt} = \vec{v}(\vec{x}(t), t) \tag{13.1}$$

where $\vec{x}$ is the particle position at a time t, $d\vec{x}/dt$ is the rate of change of the particle position in time and $\vec{v}(\vec{x}, t)$ is the 3-D velocity field generated by the model. This equation can be solved using any method suitable for solving coupled sets of nonlinear ODE's. One commonly used class of algorithms is the explicit Runge-Kutta (ERK) multi-step methods which are derived from solving the discrete integral:

$$\vec{x}(t) = \vec{x}(t_n) + \int_{t_n}^{t} \vec{v}(\vec{x}(t), \tau) d\tau . \tag{13.2}$$

Assume that $\vec{x}_n = \vec{x}(t_n)$ is the position of a particle at time $t = t_n$, then the new position $[\vec{x}_{n+1} = \vec{x}(t_{n+1})]$ of this particle at time $t = t_{n+1}(= t_n + \Delta t)$ can be determined by the 4th-order 4-stage ERK method as follows:

$$\begin{aligned}
\vec{\xi}_1 &= \vec{x}_n \\
\vec{\xi}_2 &= \vec{x}_n + \frac{1}{2}\Delta t \vec{v}(\vec{\xi}_1) \\
\vec{\xi}_3 &= \vec{x}_n + \frac{1}{2}\Delta t \vec{v}(\vec{\xi}_2) \\
\vec{\xi}_4 &= \vec{x}_n + \Delta t \vec{v}(\vec{\xi}_3) \\
\vec{x}_{n+1} &= \vec{x} + \Delta t[\frac{\vec{v}(\vec{\xi}_1)}{6} + \frac{\vec{v}(\vec{\xi}_2)}{3} + \frac{\vec{v}(\vec{\xi}_3)}{3} + \frac{\vec{v}(\vec{\xi}_4)}{6}]
\end{aligned} \tag{13.3}$$

where Δt is the time step. In this calculation, the dependence of the velocity field on time has been eliminated since the velocity field is considered stationary during the tracking time interval of Δt. It is important to understand that in a multidimensional system, the local functional derivative of $\vec{v}$ must be evaluated at the correct sub-stage point $\vec{x}$ in (x, y, z) space.

On a 2-D (x,y) plane, for example, a particle can be tracked by solving the x and y velocity equations given as

$$\frac{dx}{dt} = u, \quad \frac{dy}{dt} = v. \tag{13.4}$$

The 4-stage ERK algorithm can be written out as follows:

$$\begin{cases} \xi_1 = x_n \\ \eta_1 = y_n \end{cases} \tag{13.5}$$

$$\begin{cases} \xi_2 = x_n + \frac{1}{2}\Delta t u(\xi_1, \eta_1) \\ \eta_2 = y_n + \frac{1}{2}\Delta t v(\xi_1, \eta_1) \end{cases} \tag{13.6}$$

$$\begin{cases} \xi_3 = x_n + \frac{1}{2}\Delta t u(\xi_2, \eta_2) \\ \eta_3 = y_n + \frac{1}{2}\Delta t v(\xi_2, \eta_2) \end{cases} \tag{13.7}$$

$$\begin{cases} \xi_4 = x_n + \Delta t u(\xi_3, \eta_3) \\ \eta_4 = y_n + \Delta t v(\xi_3, \eta_3) \end{cases} \tag{13.8}$$

$$\begin{cases} x_{n+1} = x_n + \Delta t[\dfrac{u(\xi_1, \eta_1)}{6} + \dfrac{u(\xi_2, \eta_2)}{3} + \dfrac{u(\xi_3, \eta_3)}{3} + \dfrac{u(\xi_4, \eta_4)}{6}] \\ y_{n+1} = y_n + \Delta t[\dfrac{v(\xi_1, \eta_1)}{6} + \dfrac{v(\xi_2, \eta_2)}{3} + \dfrac{v(\xi_3, \eta_3)}{3} + \dfrac{v(\xi_4, \eta_4)}{6}] \end{cases}. \tag{13.9}$$

In 3-D (x,y,σ) space, a particle can be tracked by solving the x, y, and z velocity equations

$$\frac{dx}{dt} = u, \quad \frac{dy}{dt} = v, \quad \frac{d\sigma}{dt} = \frac{\varpi}{H + \zeta}, \tag{13.10}$$

where u, v, and ϖ are the x, y, and σ velocity components. The relation between ϖ and w is defined as

$$\varpi = w - (2 + \sigma)\frac{d\zeta}{dt} - \sigma\frac{dH}{dt}, \tag{13.11}$$

where w is the vertical velocity in the z coordinate direction. Let us rewrite $\varpi/(H + \zeta)$ as $\hat{\varpi}$. The 4-stage ERK algorithm for this case can be written out as follows:

$$\begin{cases} \xi_1 = x_n \\ \eta_1 = y_n \\ \gamma_1 = \sigma_n \end{cases} \tag{13.12}$$

182

$$
\begin{cases}
\xi_2 = x_n + \dfrac{1}{2}\Delta t u(\xi_1,\eta_1,\gamma_1) \\[2mm]
\eta_2 = y_n + \dfrac{1}{2}\Delta t v(\xi_1,\eta_1,\gamma_1) \\[2mm]
\gamma_2 = \sigma_n + \dfrac{1}{2}\Delta t \hat{\varpi}(\xi_1,\eta_1,\gamma_1)
\end{cases}
\tag{13.13}
$$

$$
\begin{cases}
\xi_3 = x_n + \dfrac{1}{2}\Delta t u(\xi_2,\eta_2,\gamma_2) \\[2mm]
\eta_3 = y_n + \dfrac{1}{2}\Delta t v(\xi_2,\eta_2,\gamma_2) \\[2mm]
\gamma_3 = \sigma_n + \dfrac{1}{2}\Delta t \hat{\varpi}(\xi_2,\eta_2,\gamma_2)
\end{cases}
\tag{13.14}
$$

$$
\begin{cases}
\xi_4 = x_n + \Delta t u(\xi_3,\eta_3,\gamma_3) \\[2mm]
\eta_4 = y_n + \Delta t v(\xi_3,\eta_3,\gamma_3) \\[2mm]
\gamma_4 = \sigma_n + \Delta t \hat{\varpi}(\xi_3,\eta_3,\gamma_3)
\end{cases}
\tag{13.15}
$$

$$
\begin{cases}
x_{n+1} = x_n + \Delta t[\dfrac{u(\xi_1,\eta_1,\gamma_1)}{6} + \dfrac{u(\xi_2,\eta_2,\gamma_2)}{3} + \dfrac{u(\xi_3,\eta_3,\gamma_3)}{3} + \dfrac{u(\xi_4,\eta_4,\gamma_4)}{6}] \\[2mm]
y_{n+1} = y_n + \Delta t[\dfrac{v(\xi_1,\eta_1,\gamma_1)}{6} + \dfrac{v(\xi_2,\eta_2,\gamma_2)}{3} + \dfrac{v(\xi_3,\eta_3,\gamma_3)}{3} + \dfrac{v(\xi_4,\eta_4,\gamma_4)}{6}] \\[2mm]
\sigma_{n+1} = \sigma_n + \Delta t[\dfrac{\hat{\varpi}(\xi_1,\eta_1,\gamma_1)}{6} + \dfrac{\hat{\varpi}(\xi_2,\eta_2,\gamma_2)}{3} + \dfrac{\hat{\varpi}(\xi_3,\eta_3,\gamma_3)}{3} + \dfrac{\hat{\varpi}(\xi_4,\eta_4,\gamma_4)}{6}]
\end{cases}
\tag{13.16}
$$

Many users have added a random walk-type process into this 3-D Lagrangian tracking code to simulate subgrid-scale turbulent variability in the velocity field. In FVCOM version 2.5, we only include the traditional tracking program as described above. The program can be run on both single and multi-processor computers. However, in the MPI parallel system, tracking many particles simultaneously with the model run on a multi-processor computer can significantly slow down computational efficiency, since particles moving from one sub-domain to another require additional information passing at each step. For this reason, we suggest that users use the offline version of the particle tracking code. We have provided an offline particle-tracking program in the FVCOM package. For users who want to know the update code of our offline program, please contact C. Chen.

Chapter 14: Data Assimilation Methods

14.1 Introduction

The most widely used techniques for data assimilation are (a) nudging, (b) optimal interpolation (OI), (c) variational methods (most notably adjoint methods), and (d) Kalman filtering (KF). Nudging, the most basic method, is used in MM5 to merge model-predicted values of physical variables directly to observations given *a priori* statistical assumptions about the model noise and errors in the observational data. The OI method uses the error covariance of the observations and model predictions to find their most likely linear combination (Lorenc, 1981). Similar to nudging, OI requires *a priori* statistical assumptions about the model noise and observational errors. Variational methods are based on control theory, in which a cost function, defined by the difference between model-derived and measured quantities, is minimized in a least-square sense under the constraint that the governing equations of the model remain satisfied (Le Dimet and Talagrand, 1986; Thacker and Long, 1988; Tziperman and Thacker, 1989; Bergamasco et al., 1993; Morrow and De Mey, 1995). In this framework, an adjoint system to the model can be constructed which can directly yield the sensitivity of this cost function to the specified control variables such as the model initial or boundary conditions. This adjoint can be derived directly from the model's governing equations. However, in practice it is more consistent to use an adjoint derived from the discrete representation of the governing equations in the model. In some cases, this discrete system adjoint can be constructed directly from the model source code using automatic differentiation (AD) methods such as OpenAD (see http://www.mcs.anl.gov/OpenAD/ and Utke et al. (2008). We have not attempted to construct an adjoint model for FVCOM yet, but hope to consider this in the near future.

Kalman Filters are the most sophisticated statistical approaches, and are commonly used for nowcasting/forecasting in ocean and atmospheric models (Evensen, 1992, 1993; Blanchet, 1997; Ghil and Malanotte-Rizzoli, 1991). Dr. Malanotte-Rizzoli at MIT and her collaborators have constructed a series of assimilation packages first applied to idealized models as proof-of-concept tests and successively applied to fully realistic, primitive equations models. They include a Reduced Rank Kalman filter (RRKF) (Buehner and Malanotte-Rizzoli, 2003), an Ensemble Kalman Filter (EnKF) (Zang and

Malanotte-Rizzoli, 2003), deterministic and stochastic ensemble Kalman filters (Lyu et al, 2005a), and an Ensemble Transform Kalman Filter (ETKF) (Lyu et al, 2005b). The EnKF has shown advantages in dealing with strongly nonlinear systems which, we believe, is most relevant to the coastal ocean. The ETKF was introduced by Bishop et al. (2001) for optimally deploying adaptive observations and is presently the most used approach in meteorology. Lyu et al (2005a,b) have applied both RRKF and ETKF to design optimal fixed and adaptive observational arrays in an idealized model of the wind-driven circulation in a double gyre ocean, with the ultimate objective to extend to a fully realistic Ocean General Circulation Model. Kalman Filters can be used to design optimal observational arrays which give the minimum trace of the forecast error covariance over the region of interest. In particular, the filters can be used to conduct sensitivity studies of the effectiveness of the optimal fixed/adaptive network to improve model forecasts with respect to a) number and type of observations, b) targeting different regions characterized by different dynamics and energetics, c) duration of the targeted (adaptive) lead time corresponding to successive assimilations, and d) ensemble size in a filter among other factors.

Several data assimilation modules have been developed for FVCOM. They include 3-D nudging, OI, and a suite of Kalman Filters. FVCOM currently has data-assimilation capabilities for hindcast skill improvement which utilize the 3-D nudging and Kalman Filters. The nudging method is fast and practical for forecast applications, but it is prone to causing unphysical behavior if the data coverage is too coarse or the associated parameters are not properly set. OI requires the covariance of variables, which are usually hard to estimate due to the scarcity of required data. Recently we have built a covariance map of SST for the Gulf of Maine, which can be used to apply OI for SST assimilation. In collaboration with P. Malanotte-Rizzoli, we have implemented a Kalman Filter module into FVCOM. This set includes the 1) Reduced Rank Kalman Filter (RRKF), 2) Ensemble Kalman Filter (EnKF), 3) Ensemble Transform Kalman Filter (ETKF), 4) Ensemble Square-Root Kalman Filter (EnSKF), and 5) singular evaluative interpolated Kalman filter (SEIK) for hind- and now-casting applications. This module has been validated for a number of idealized test cases and simplified ocean cases. Chen et al. (2009b) first described the RRKF, EnKF, and EnSKF formulations in FVCOM and

results of three test cases: a) an annular-shaped constant-depth shelf driven by tidal forcing at the open boundary, b) a linear slope continental shelf with river discharge, and c) a rectangular estuary with tidal flushing intertidal zones and freshwater discharge. Xue et al. (2011) then described the application of EnKF and EnTKF for conducting simple observing system simulation experiments (OSSEs) in Nantucket Sound (MA). They also compared the SEIK with EnKF for assimilation efficiency. Recently Xue et al (2012) used the EnKF to conduct OSSEs to identify optimal monitoring sites for dissolved oxygen sampling in Massachusetts Bay (MA).

A brief description of Nudging and OI is given here. The Kalman Filters involve complex mathematic equations and the details of these equations were given in the set of published papers by Malanotte-Rizzoli and her collaborators referenced above. Additional information about how they are implemented and work in FVCOM can be found in Chen et al. (2009b) (see paper and supplemental material at http://www.agu.org/journals/jc /jc0905/2007JC004548/), Xue et al. (2011), and Xue et al. (2012).

14.2 The Nudging Method

Let $\alpha(x,y,z,t)$ be a variable selected to be assimilated and $F(\alpha,x,y,z,t)$ represents the sum of all the terms in the governing equation of $\alpha(x,y,z,t)$ except the local temporal change term, then the governing equation of $\alpha(x,y,z,t)$ with inclusion of nudging assimilation is given as

$$\frac{\partial \alpha(x,y,z,t)}{\partial t} = F(\alpha,x,y,z,t) + G_\alpha \frac{\sum_{i=1}^{N} W_i^2(x,y,z,t)\gamma_i(\alpha_o - \hat{\alpha})_i}{\sum_{i=1}^{N} W_i(x,y,z,t)} \qquad (14.1)$$

where α_o is the observed value; $\hat{\alpha}$ is the model-predicted value; N is the number of observational points within the search area; γ_i is the data quality factor at the ith observational point with a range from 0 to 1; and G_α is a nudging factor that keeps the nudging term to be scaled by the slowest physical adjustment process. The selection of G_α must satisfy the numerical stability criterion given by

$$G_\alpha < \frac{1}{\Delta t} \tag{14.2}$$

Normally, G_α is set to approximately the magnitude of the Coriolis parameter. $W_i(x,y,z,t)$ is a product of weight functions given as

$$W_i(x,y,z,t) = w_{xy} \cdot w_\sigma \cdot w_t \cdot w_\theta \tag{14.3}$$

where w_{xy}, w_σ, w_t, and w_θ are horizontal, vertical, temporal and directional weighting functions, respectively. The mathematical expressions of these functions are given as

$$w_{xy} = \begin{cases} \dfrac{R^2 - \hat{r}^2}{R^2 + \hat{r}^2}, & 0 \le \hat{r} \le R \\ 0, & \hat{r} > R \end{cases} \tag{14.4}$$

$$w_\sigma = \begin{cases} 1 - \dfrac{|\sigma_{obs} - \sigma|}{R_\sigma}, & |\sigma_{obs} - \sigma| \le R_\sigma \\ 0, & |\sigma_{obs} - \sigma| > R_\sigma \end{cases} \tag{14.5}$$

$$w_t = \begin{cases} 1, & |t - t_o| < T_w/2 \\ \dfrac{T_w - |t - t_o|}{T_w/2}, & T_w/2 \le |t - t_o| \le T_w \\ 0, & |t - t_o| > T_w \end{cases} \tag{14.6}$$

$$w_\theta = \frac{\big||\Delta\theta| - 0.5\pi\big| + c_1\pi}{(0.5 + c_1)\pi} \tag{14.7}$$

where R is the search radius, $\hat{r}$ is the distance from the location where the data exists, R_σ is the vertical search range, T_w is half of the assimilation time window, and $\Delta\theta$ is the directional difference between the local isobath and the computational point with c_1 a constant ranging from 0.05 to 0.5.

14.3 The OI Method

Optimal interpolation (OI) is an alternative simple data assimilation method similar to the nudging method. A detailed comparison of OI with other data assimilation methods was given in detail in Kantha and Clayson (2000) and a brief description of this scheme is given here for users who are not familiar with this scheme.

Let X_f, X_a, and X_o be the model forecast, assimilated (analysis) and observed values of a model variable X, respectively, and assume that they satisfy a linear relationship given as

$$X_a = X_f + \sum_{k=1}^{M} a_k (X_{o,k} - X_{f,k})$$

(14.8)

where M is the total data points involved in the optimal interpolation for X at a node point. Defining that the true value of X is X_T at the assimilated node, $X_{T,k}$ at the kth observed point, $e_a = X_a - X_T$, $e_f = X_f - X_T$, $e_{o,k} = X_{o,k} - X_{T,k}$, and $e_{f,k} = X_{f,k} - X_{T,k}$, the analysis error e_a is equal to

$$e_a = e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})$$

(14.9)

The analysis error covariance $P_a = e_a^2$, which is given as

$$P_a = [e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})][e_f + \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})]$$

$$= e_f^2 + 2\sum_{k=1}^{M} a_k (e_f e_{o,k} - e_f e_{f,k}) + [\sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k})]^2$$

(14.10)

In the least square fitting method, the error in e_a must be a minimum when the first differentiation condition of $\partial P_a / \partial a_k = 0$ is satisfied, i.e.

$$\begin{cases} (e_{o,1} - e_{f,1}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,1} - e_f e_{o,1}) \\ (e_{o,1} - e_{f,2}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,2} - e_f e_{o,2}) \\ \vdots \\ \vdots \\ (e_{o,M} - e_{f,M}) \sum_{k=1}^{M} a_k (e_{o,k} - e_{f,k}) = (e_f e_{f,M} - e_f e_{o,M}) \end{cases}$$

(14.11)

Assuming that $e_{f,k}$ is not correlated with $e_{o,k}$, eq. (14.11) can be simplified to

$$\begin{cases} \sum_{k=1}^{M} a_k (e_{o,k} e_{o,1} + e_{f,k} e_{f,1}) = e_f e_{f,1} \\ \sum_{k=1}^{M} a_k (e_{o,k} e_{o,2} + e_{f,k} e_{f,2}) = e_f e_{f,2} \\ \vdots \\ \vdots \\ \sum_{k=1}^{M} a_k (e_{o,k} e_{o,M} + e_{f,k} e_{f,M}) = e_f e_{f,M} \end{cases} \qquad (14.12)$$

or

$$\begin{cases} (e_{o,1}^2 + e_{f,1}^2) a_1 + (e_{o,2} e_{0,1} + e_{f,2} e_{f,1}) a_2 \cdots + (e_{o,M} e_{o,1} + e_{f,M} e_{f,1}) a_M = e_f e_{f,1} \\ (e_{o,1} e_{o,2} + e_{f,1} e_{f,2}) a_1 + (e_{o,2}^2 + e_{f,2}^2) a_2 \cdots + (e_{o,M} e_{o,2} + e_{f,M} e_{f,2}) a_M = e_f e_{f,2} \\ \vdots \\ \vdots \\ (e_{o,1} e_{o,2} + e_{f,1} e_{f,2}) a_1 + (e_{o,2} e_{o,M} + e_{f,2} e_{f,M}) a_2 \cdots + (e_{o,M}^2 + e_{f,M}^2) a_M = e_f e_{f,M} \end{cases} \qquad (14.13)$$

This can be written in matrix form as

$$\hat{P} \cdot \hat{a} = \hat{f} \qquad (14.14)$$

where

$$\hat{P} = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1M} \\ P_{21} & P_{22} & \cdots & P_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ P_{M1} & P_{M2} & \cdots & P_{MM} \end{pmatrix}; \ \hat{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{pmatrix}; \ \hat{f} = \begin{pmatrix} e_f e_{f,1} \\ e_f e_{f,2} \\ \vdots \\ e_f e_{f,M} \end{pmatrix} \qquad (14.15)$$

and

$$P_{i,k} = e_{o,i} e_{o,k} + e_{f,i} e_{f,k}, \ i=1, 2,\ldots, M; \ k=1,2,\ldots, M. \qquad (14.16)$$

When the observational and forecast error covariance values are known or specified, parameter a_k can be determined by using a state-of-the-art linear algebraic equation solver to solve (14.14).

In real applications, for simplification, we can assume that the observational errors are zero and the forecast error covariance satisfies a normal distribution given by

$$P_{ik} = e^{-(\frac{r_{ik}}{d})^2} \qquad (14.17)$$

where r_{ik} is the horizontal distance between i and k points and d is the correlation radius. With this approach, the OI scheme should be very similar to the nudging data assimilation scheme.

The nudging and OI data assimilation methods are practical approaches for the purpose of model application to the real-time simulation and assimilation. However, they lack rigorous scientific support and are not generally useful for sensitivity studies of model parameters.

14.4. The Kalman Filters

Let x^t be an array of the true values, x^f be an array of the forecast values, x^a an array of analysis values, and y an array of observational values. We can define that

Analysis error: $\qquad e_a = x^a - x^f$ $\qquad\qquad\qquad\qquad$ (14.18)

Forecast error: $\qquad e_f = x^t - x^f$ $\qquad\qquad\qquad\qquad$ (14.19)

Observational error: $\quad e_o = y - x^f$ $\qquad\qquad\qquad\qquad$ (14.20)

The forecast error covariance P and the observational error covariance R thereafter can be defined as

$$P^f = \overline{e_f e_f^T}, \quad R = \overline{e_o e_o^T} \qquad\qquad (14.21)$$

In a forecast model, the value of $\boldsymbol{x}^f$ at time step i can be predicted by

$$x^f(i) = M_{i-1 \to i}[x^a(i-1)] \qquad\qquad (14.22)$$

where M presents the nonlinear model operator. The forecast error covariance is equal to

$$P^f = M_{i-1 \to i} P^a(i-1) M_{i-1 \to i}^T + Q(i-1) \qquad\qquad (14.23)$$

where $Q(i-1)$ is the system error covariance matrix. In the Kalman filter forecast model system, the analysis values x^a is calculated by

$$x^a(i) = x^f(i) + K(i)[y(i) - Hx^f(i)] \qquad\qquad (14.24)$$

where K is the Kalman gain matrix, which is equal to

$$K(i) = P^f(i)H^T[HP^f(i)H^T + R(i)]^{-1} \qquad\qquad (14.25)$$

and H is an observation operator that functions as an objective map to interpolate the model data onto the observational points. The analysis error covariance is given as

$$P^a(i) = [I - K(i)]P^f(i). \qquad\qquad (14.26)$$

In general, the size of the covariance matrix is huge. For example, P^f : $[N \times N]$ and M : $[N \times N]$, and N can be of $O(10^6 - 10^7)$, which makes it impractical to use this method on most computers.   For this reason, a family of Kalman Filters (Reduced Rank Kalman Filter, Ensemble Kalman Filter, Ensemble Square Root Kalman Filter and Ensemble Transform Kalman Filter) have been developed that require less computational power than the original filter. A brief description of these other Kalman Filters and how they are implemented in FVCOM is given below.

14.4.1 Reduced Rank Kalman Filter (RRKF)

Let x^f be an array with a dimension of N (FVCOM output and RRKF input); x^a an array with a dimension of N (that are output from RRKF to use to refresh the initial conditions for FVCOM input); y an array with a dimension of N_o (that is an input for RRKF); E_r the resolved empirical orthogonal functions (EOFs) with dimensions of $N \times N_e$ (the stationary input of RRKF); K_r the stationary Kalman gain in the reduced space of N_e
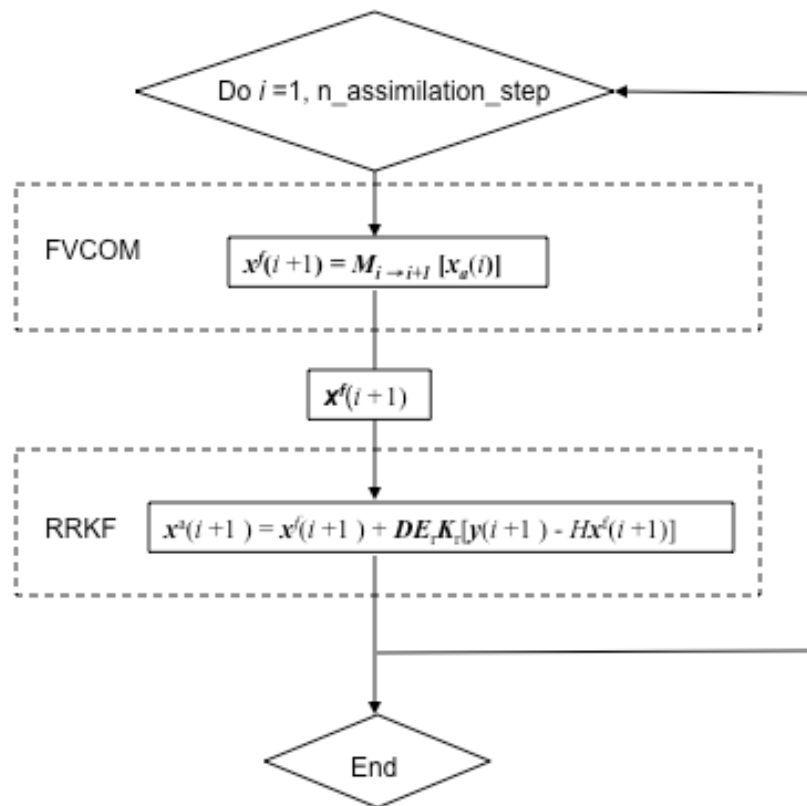
Fig. 14.1: Schematic of the implementation of RRKF into FVCOM

$\times N_o$] (the stationary input of RRKF (stationary input; N_e the dimension of the EOFs subspace; i and $i+\Delta T$ two subsequent assimilation time steps of FVCOM; ΔT the assimilation interval; M_i the forward FVCOM model with initial conditions specified by the analysis solution; H an observation operator that functions as an objective map to interpolate the model data onto the observational points; and D is the spatially averaged standard deviation of each variable. The flow chart of RRKF in FVCOM is illustrated in Fig. 14.1.

A detailed description of RRKF was given in Buehner and Malanotte-Rizzoli (2003). S. Lyn, who worked with P. Malanotte-Rizzoli as a postdoctoral investigator at MIT, helped us implement RRKF into FVCOM. He worked together with P. Xue, Q. Xu and Z. Lai in testing the RRKF code in idealized cases.

The RRKF is developed based on linear theory. In a linear system, if the observational network, observational and model error covariances are stationary and all neutrally stable and unstable modes are measurable, the forecast error covariance reaches an asymptotically stationary result and then a stationary Kalman gain (K_r) can be derived efficiently by the doubling algorithm (Anderson and Moore 1979). This stationary K_r is calculated from the control model run and it is applied in the data assimilation process of the RRKF.

The RRKF in FVCOM is operated following the procedure given below:

Step 1: Determination the number of resolved EOFs (E_r) from the control model run:

$$D^{-1} \hat{X}_f \hat{X}_f^T D^{-1} = E \Lambda E^T \tag{14.18}$$

where $\hat{X}_f = x^f - \bar{x}^f$, $\bar{x}^f$ the mean of x^f, E the EOF matrix; and Λ is the diagonal covariance eigenvalue matrix.

Step 2: Linearization of the model in the resolved EOF subspace (build M_r from E_r):

$$M_{\hat{i}} = \frac{1}{\alpha} E_r^T D^{-1} [M(x_0 + \alpha D e_i) - M(x_0)] \tag{14.19}$$

where M presents the nonlinear model; subscripts "r" and "i" of M denote the linearized model in the resolved subspace and the ith column of M_r; α is the perturbation size; e_i the ith retained EOF; and x_0 is the specified time mean of a long model run without assimilation.

Step 3: Projection of the error covariance into the resolved EOFs subspace and estimation of the model and observation errors:

$$P_r^f = E_r^T P^f E_r; \quad P_r^a = E_r^T P^\alpha E_r; \quad M_r = E_r^T M E_r; \quad Q_r = \gamma \Lambda \tag{14.20}$$

$$R = R_m + H_u P_u^f H_u^T \tag{14.21}$$

where P_r^a is the analysis error covariance matrix in the resolved subspace; Q_r the 'pesudo' model error covariance; R the observational error covariance; R_m the actual measurement error; and P_r^f the forecast error covariance matrix in the resolved subspace.

Step 4: Calculation of the stationary Kalman gain K_r in the reduced subspace by the doubling algorithm, estimation of the difference between the observations and forecast and projection to the full space by multiplying E_r

$$K_r(t) = P_r^f(t)(H_r P_r^f(t) H_r^t + R)^{-1} \tag{14.22}$$

$$P_r^a(t) = (I - K_r(t) H_r) P_r^f(t) \tag{14.23}$$

$$P_r^f(t+1) = M_r P_r^a(t) M_r^T + Q_r \tag{14.24}$$

where $H_r = HDE_r$ and P_r^f is asymptotically stationary as t→∞ if H, R and Q is stationary with linear dynamics.

Step 5: Data assimilation by using a stationary Kalman gain K_r

$$x^a(t) = x^f(t) + DE_r K_r[y(t) - Hx^f(t)] \tag{14.25}$$

RRKF works efficiently in a linear system but not for a nonlinear system. We tested it for various idealized cases such as tidal waves in the circular lakes, the flooding/drying process in the rectangular shape estuary. It produces a fast convergence solution for the linear tidal wave case, but never converges in the estuarine case characterized with nonlinear dynamics.

14.4.2. Ensemble Kalman Filter (EnKF)

Evensen (1994) suggested that the error covariance relative to the mean of ensemble model results could provide a better estimation of the error covariance defined in the classical Kalman Filter. The EnKF is constructed by running a forecast model driven by a set of initial conditions and then estimate the error covariance relative to the ensemble

mean to determine the ensemble analysis values for the next time step forecast. This approach is illustrated in Fig. 14.2.
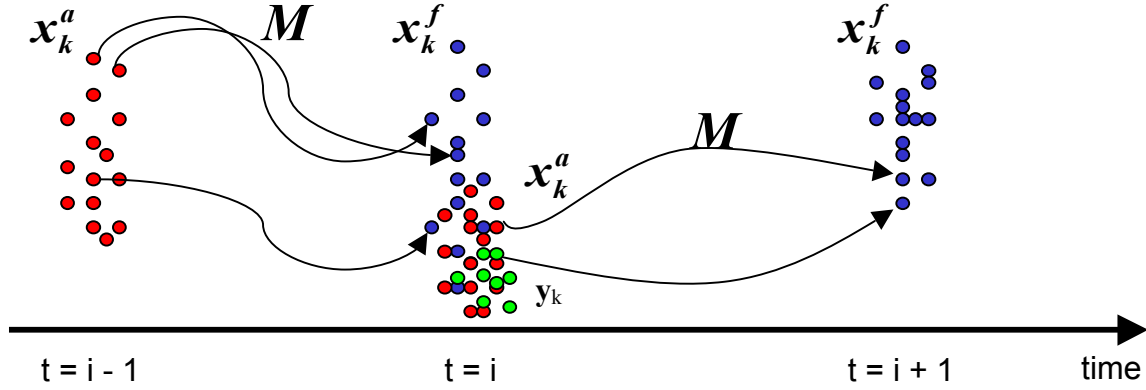
Fig. 14.2: The illustration of the forecast model run with EnKF. This figure was drawn by S. Lyn.

Let k denote the kth ensemble and N_E the total number of the ensemble members selected in the forecast model run. The forecast value at time step i for the kth ensemble model run can be estimated by

$$x_k^f(i) = M_{i-\Delta t \to i}[x_k^a(i-1)], \quad k = 1,2,...,N_e \tag{14.26}$$

and the analysis values at time step i for the kth ensemble model run are calculated b

$$\mathrm{x}_k^a(i) = \mathrm{x}_k^f(i) + \mathrm{K}(i)[\mathrm{y}_k(i) - \mathrm{Hx}_k^f(i)], \quad k = 1,\ 2,....,N_e \tag{14.27}$$

Define that

$$X_f = \left\{ [x_k^f - \bar{x}^f]/\sqrt{N_e - 1} \right\} \quad k = 1,2,....,N_e \tag{14.28}$$

then the forecast error covariance can be estimated by

$$P^f(i) \approx X_f(i)X_f^T(i): \quad [N \times N_e][N_e \times N] \tag{14.29}$$

The Kalman gain is equal to

$$K(i) = X_f X_f^T H^T (H X_f X_f^T H^T + R)^{-1} \tag{14.30}$$

To conduct the EnKF, we need to create an ensemble of the observational data constructed with the perturbation relative to the real value, *i. e.*,

$$y_k = y + \delta_k, \quad \delta_k = N(0, \sqrt{R}) \tag{14.31}$$

where

$$R = \overline{\delta\delta^T} \tag{14.32}$$

In the situation with a sufficiently large number of ensembles, the ensemble analysis error covariance matrix can be updated with a relationship as

$$P_e^a(t) = (I - K_r(t)H)P_e^f(t) \tag{14.33}$$

This will give us optimal analysis values in the maximum likelihood sense and in the
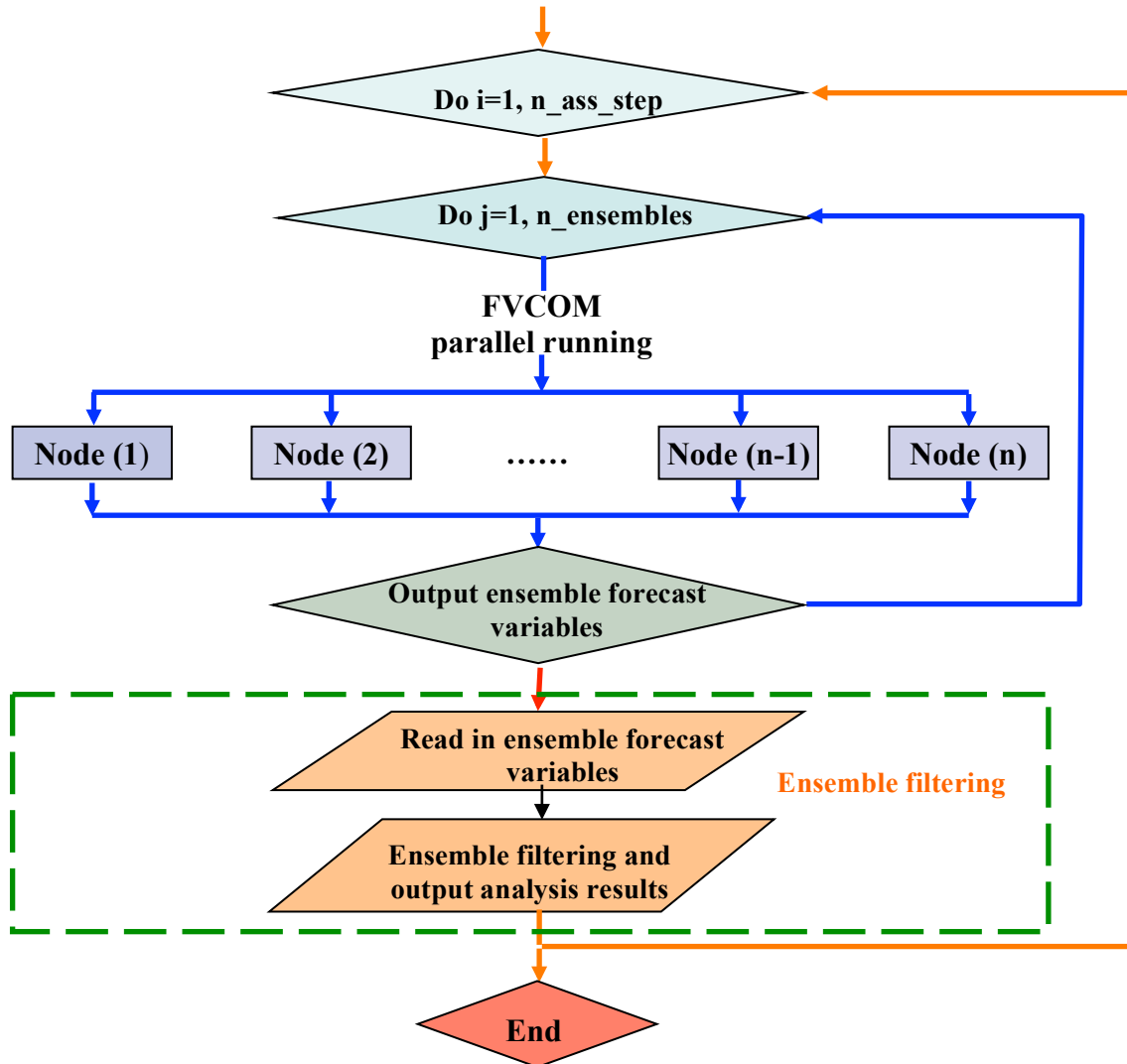
Fig. 14.3: Flow chart schematic of EnKF implemented in FVCOM.

minimum variance sense. In the situation with a small number of ensembles, the perturbed observations required by EnKF may cause a rank deficiency problem for the estimation of P^f and an underestimate of P^a, which leads to filter divergence (Whitaker and Hamill, 2002). The solution is to conduct EnKF with covariance localization (Houterkamer and Mitchell, 1998) and covariance inflation (Wang and Bishop, 2003).

The flow chart schematic of EnKF implemented in FVCOM is shown in Fig. 14.3. EnKF is developed for the full nonlinear system. For a linear system, RRKF works well and also fast, but it sometimes fails to resolve linear waves in the idealized, linear coastal ocean system. In such a case, EnKF works well.

14.4.3. Ensemble Square-Root Kalman Filter (EnSRF)

EnSKF is a stochastic filter that requires perturbed sets of observational values. In this system, the perturbed observational values are usually constructed by a control observation in addition to random noise sampled from the assumed observational error distribution. There are derivatives of EnKF that are conducted with deterministic observational ensembles. These filters are known as the EnSRF (Whitaker and Hamill, 2002), Ensemble Transform Kalman Filter (ETKF) (Bishop et al., 2001) and the Ensemble Adjustment Kalman Filter (EAKF) (Anderson, 2001). Actually, these filters are a family of the deterministic square root filter (Tippett et al., 2003). A brief description of one type of EnSRF is given below.

Assuming that the forecast and observational error covariance is Gaussian distributed, the ensemble Kalman Filter provides optimal analysis values which satisfy the classical Kalman Filter covariance form as

$$P_e^a = (1 - K_r H) P_e^f \tag{14.34}$$

and the Kalman gain is

$$K_e = P_e^f H^T [H P_e^f H^T + R_e]^{-1}. \tag{14.35}$$

Eq. (14.34) can be rewritten into

$$P_e^a = X_a X_a^T \tag{14.36}$$

where

$$X_a = X_f \hat{T} \tag{14.37}$$

and $\hat{T}$ is the square root matrix defined as

$$\hat{T} = \{I - X_f H^T [H X_f X_f^T H^T + R]^{-1} H X_f\} \tag{14.47}$$

where I is $N_e \times N_e$ unit matrix. In this case, an ensemble of the analysis deviation X_a can be estimated deterministically from an ensemble of the forecast deviation, and also the analysis ensemble has a desired error covariance.

Assuming that the observation errors are uncorrelated, the observations can be assimilated serially. In this case, $HX_f X_f^T H^T + R$ in (14.47) is simplified to be a scalar in the case of a single observation. Then the square root matrix can be easily calculated as

$$\hat{T} = [1 - \beta X_f^T H^T H X_f][1 - \beta X_f^T H^T H X_f]^T \tag{14.48}$$

where $\beta = [a + \sqrt{Ra}]^{-1}$.

EnSRF does not require a perturbed observation set, so that the ensemble mean of analysis values $\overline{x}^a$ can be updated directly by the mean of forecast ensemble produced from a traditional Kalman Filter equation as

EnSRF does not require a perturbed observation set, so that the ensemble mean of analysis values $\overline{x}^a$ can be updated directly by the mean of forecast ensemble produced from a traditional Kalman Filter equation as

$$\overline{x}^a(i) = \overline{x}^f(i) + X_f X_f^T H^T [HX_f X_f^T H^T + R]^{-1}(y - H\overline{x}^f) \tag{14.49}$$

and each analysis member x_k^a can be calculated by

$$\overline{x}_k^a(i) = \overline{x}^a(i) + X_k^a \sqrt{N_e - 1} \quad k = 1, 2, \ldots\ldots N_e \tag{14.50}$$

The flow-chart schematic for EnSRF in FVCOM is the same as Fig. 14.3.

14.4.4. Ensemble Transform Kalman Filter (ETKF)

The ETKF is very similar to EnSRF with a linear transformation of the forecast perturbation into the analysis perturbation by $\hat{T}$

$$X_a = X_f \hat{T} \tag{14.51}$$

Bishop et al. (2001) proposed the form of the transformation matrix *T* as:

$$\hat{T} = C(\tilde{A} + I)^{-1/2} \tag{14.52}$$

where columns of the matrix *C* contain the eigenvectors of $X_f^T H^T R^{-1} H X_f$ and $\tilde{A}$ is the nonzero diagonal matrix that satisfies a relationship with *C* as

$$X_f^T H^T R^{-1} H X_f = C\tilde{A}C^T \tag{14.53}$$

In this case,

$$x^a(i) = \overline{x}^f(i) + X_f C \tilde{A}^{1/2}(\tilde{A} + I)^{-1} E^T \{R^{-1/2}y - H\overline{x}^f\} \tag{14.54}$$

where

$$E = HX_f C\tilde{A}^{-1/2} \tag{14.55}$$

In FVCOM, ETKF is used for adaptive observation optimization in which X_a is used to evaluate the analysis ensemble error covariance under different observational strategies. This approach allows us to use the model to determine the optimal observational network for a selected region.

14.4.5. The Validation Experiments

The FVCOM development team has worked with P. Malanotte-Rizzoli at MIT to validate a package of Kalman Filters implemented into FVCOM. The test problems include tidal oscillations in a flat-bottom circular basin, tidal flooding/draining process in an idealized estuary with an intertidal zone; river discharge plume over an idealized continental shelf, and also adaptive field measurement designs using the ETKF for the plume case. Since we have not yet published any results, we include only a brief summary here.

Test problem: Tidal Oscillations in a Flat-Bottom Circular Basin

Conditions: linear, 2-D, no friction, shallow water, the radius of the basin R = 50 km, M_2 tidal forcing at the open boundary. Case I: normal oscillation in which water depth H
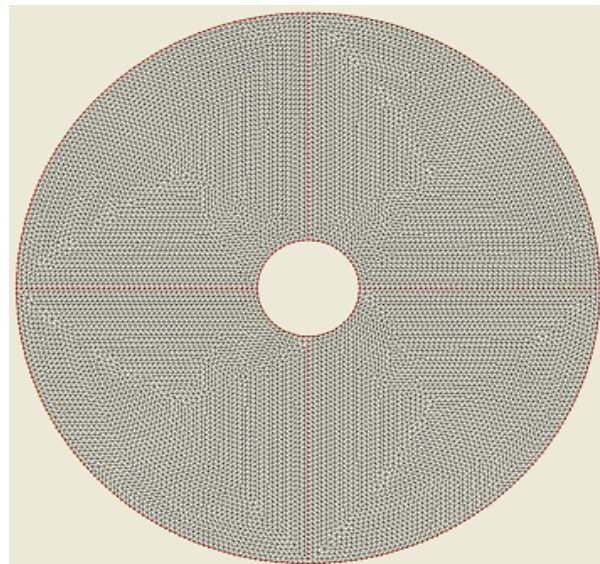
Fig. 14.4: Illustration of the circular basin and unstructured grid.

= 10 m, M_2 tidal elevation at the OB is given as 1.0 m. Case II: near-resonance in which H = 1 m, M_2 tidal elevation at the OB is given as 1 mm. See Fig. 14.4 for the computational domain and triangular grid used to configure FVCOM. The results for the near-resonance case are shown here.

In the near-resonance case, both RRKF and EnKF show a fast convergence to the true solution after a perturbation. Fig. 14.5 shows the RMS analysis results of RRKF. Only one current measurement is made and used in the filter. At 0 hour, the perturbation is generated by replacing the model-computed surface elevation and currents with zeros. RRKF quickly induces the solution to converge towards the true state in just one tidal cycle.
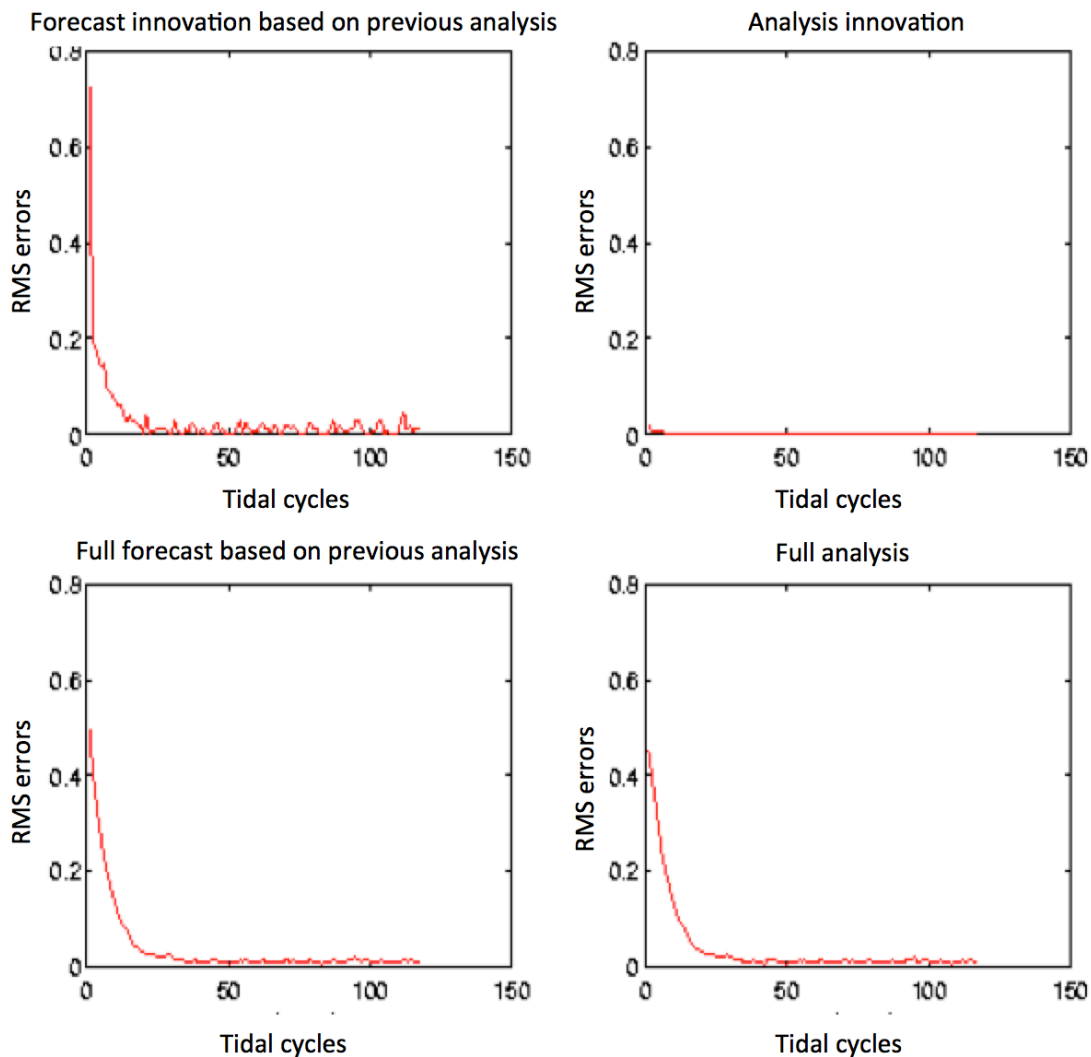
Fig. 14.5: RMS error analysis of RRKF for the near-resonance case.

Fig. 14.6 shows the surface elevation patterns for the true state, analysis state and error at 0 hour (when the perturbation is generated), 1 hour later, and then 12 hours after the perturbation was generated. Even with this extreme initial perturbation, RRKF works quickly in this case to return the solutions back to the true state.
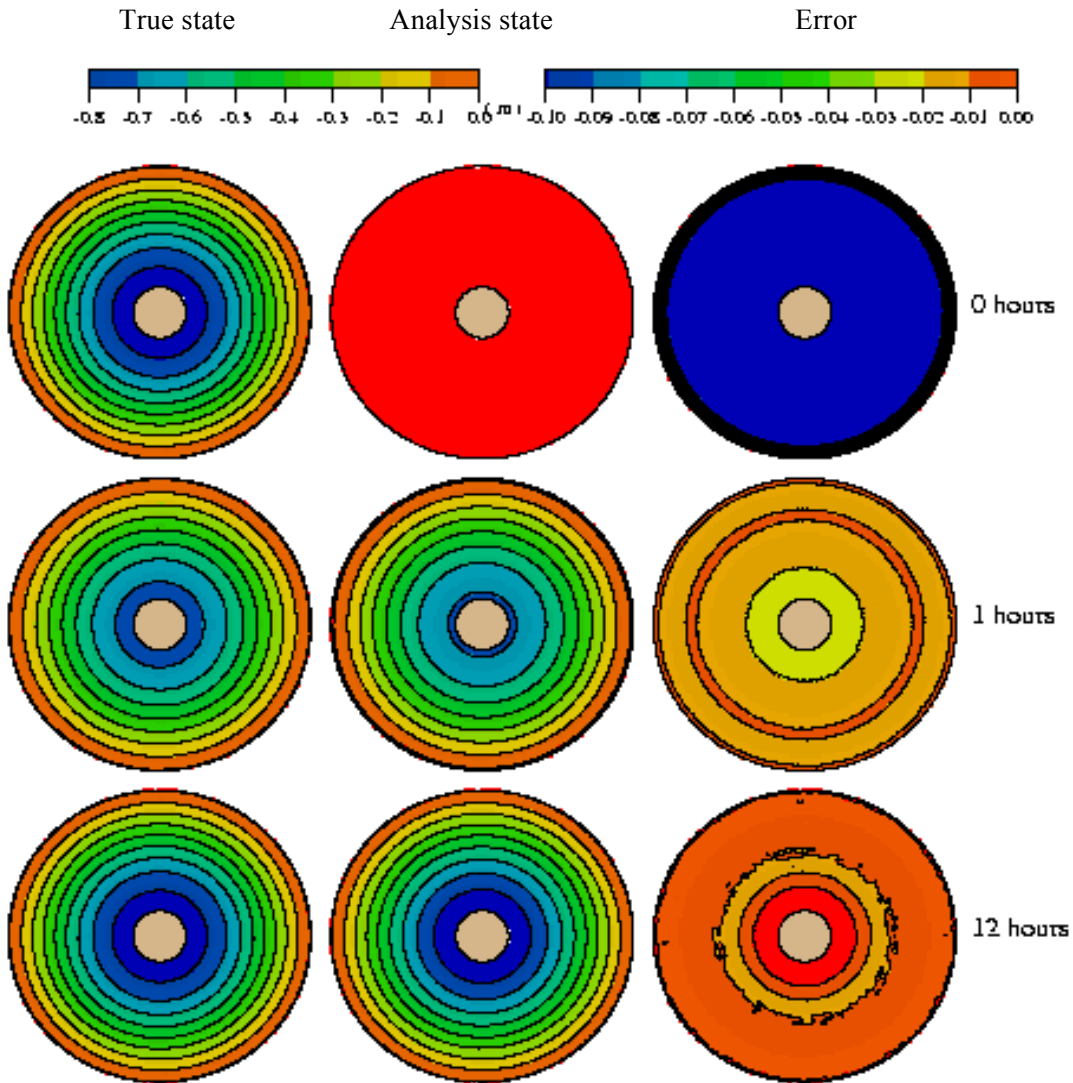
Fig. 14.6: The distributions of the surface elevation of true state (left), analysis state (middle), and error (right) at 0 hour (when perturbation is created), 1 hour and 12 hours for the RRKF.

In the EnKF experiment, 20 ensembles were chosen and one surface elevation measurement was taken. Fig. 14.6 shows the RMS analysis results of EnKF for the near-resonance case. Even though we only included one measurement site, the model solution quickly converged towards the true solution in less than one tidal cycle. 20 ensembles were constructed from previous time frames initialized with random perturbations.

Fig. 14.7 shows the distributions of the surface elevation for the true state, analysis state and rms errors predicted by EnKF. It can be seen that EnKF works efficiently to suppress the perturbation and direct the numerical solution back to the true state.
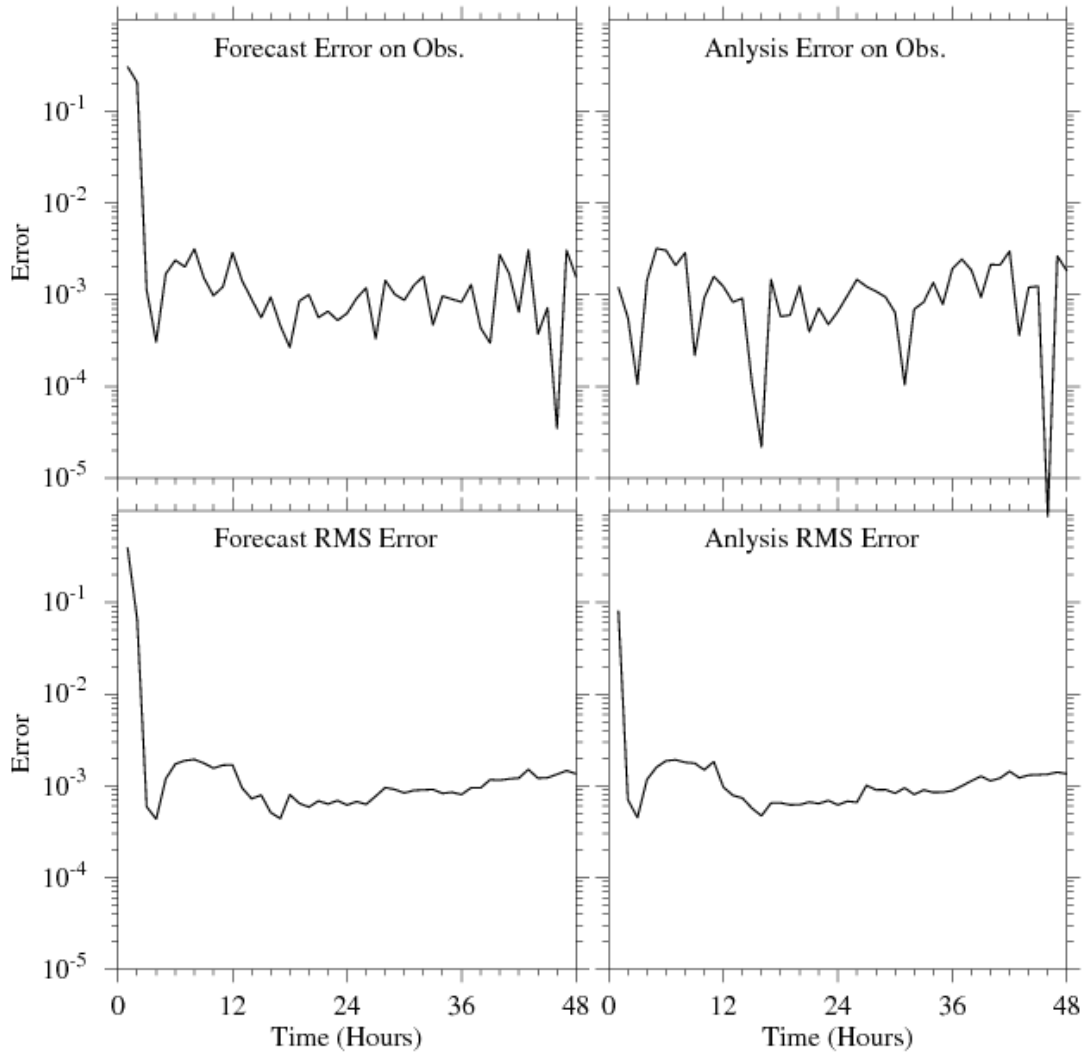
Fig. 14.7: The RMS analysis of EnKF (with 20 ensembles) for the near-resonance case.

We also did experiments to examine the locations of the best measurement sites to achieve optimal model convergence rate, as well as examine the necessary duration of a ship-tracking measurement required to achieve sufficient model convergence.

The Kalman Filters implemented in FVCOM render the model applicable for forecast/hindcast applications for the real coastal ocean.
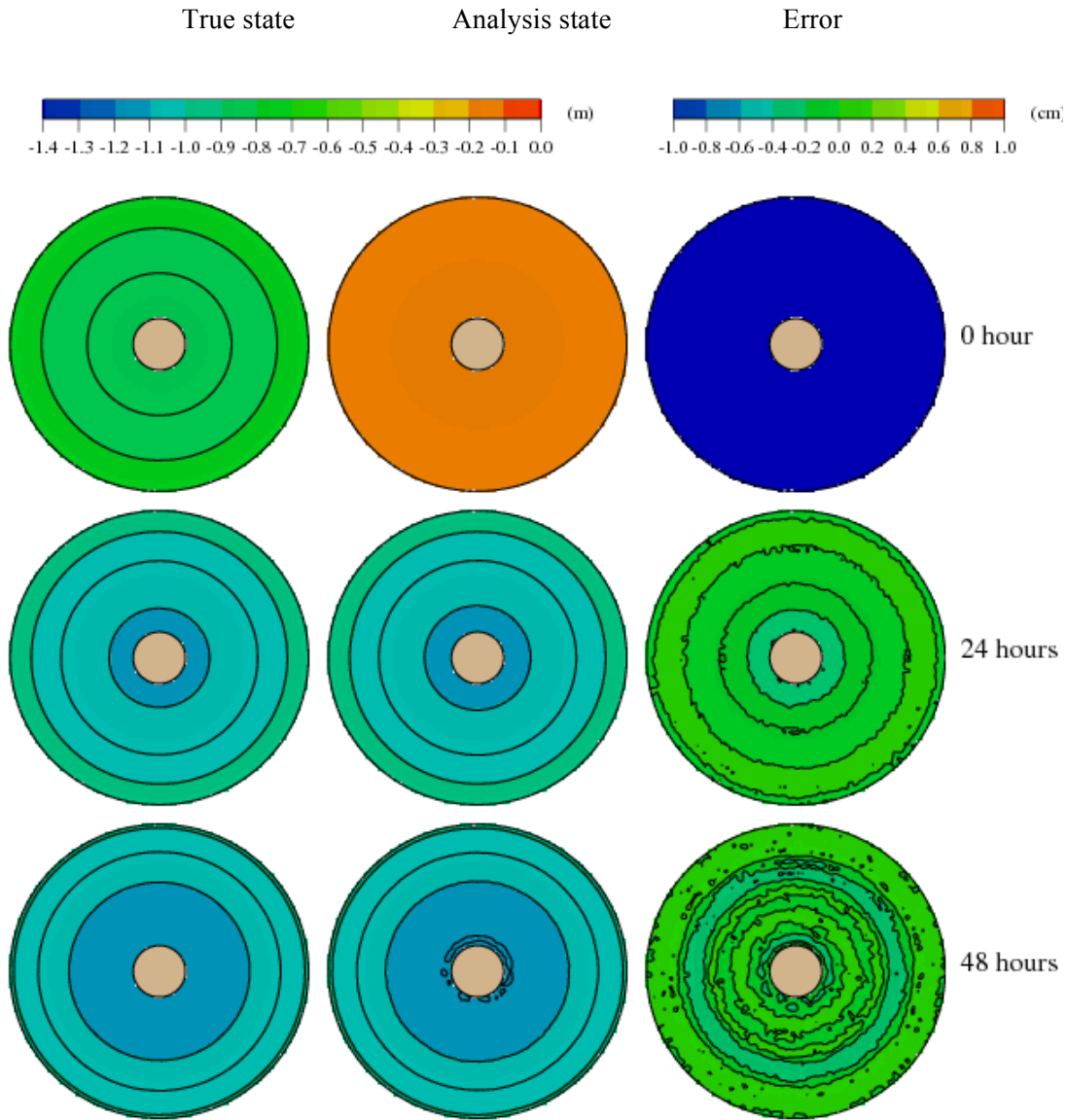
Fig. 14.8: The distributions of surface elevation of the true state (left), analysis state (middle) and error (right) at 0 hour (time of perturbation creation), 24 hours, and 48 hours later. In this case, the EnKF was used with 20 ensembles.

Chapter 15: Code Parallelization

The FVCOM code has been parallelized using a Single Processor Multiple Data (SPMD) approach. The domain is decomposed using the METIS graph partitioning libraries. The interprocessor communication is explicitly defined using Message Passing Interface (MPI) calls. The resulting implementation is highly portable and will run efficiently on a variety of parallel computer architectures including both shared and distributed memory systems. The basic elements of the parallelization are as follows.

1) *Domain Decomposition*: The domain (grid) is decomposed into N equal partitions (subdomains) where N is the number of processors to be used for the computation.

2) *Domain Setup*: Each processor sets up an FVCOM integration in its respective subdomain.

3) *Data Exchange*: During the calculation, information is exchanged between processors across subdomain boundaries to ensure correctness of the boundary fluxes.

4) *Data Collect*: Output data is collected from individual processors and reconstructed into a global array before being written to disk.

15.1. Domain Decomposition

The domain decomposition is performed using the METIS graph partitioning libraries (Karypis and Kumar, 1998). Given a list of elements, information about their connectivity, and a user input desired number of partitions, METIS is tasked to assign elements to partitions under the following constraints.

1) Each partition will contain roughly the same number of elements.

2) The total length of the boundary between partitions is to be minimized.

The first constraint pertains to the concept of load balancing. In a code such as FVCOM where the computational effort is dominated by explicit integration of the primary equations, the work required is roughly proportional to the number of elements (triangles) in a domain. Thus to ensure equal workload among the processors, the decomposition must provide the same number of elements to each partition. The second constraint is introduced to reduce communication overhead. Communication of data between processors must be performed to ensure correctness of the flux at the interprocessor

boundary. This communication represents overhead in a parallel program and directly contributes to a reduction in the efficiency of the parallel implementation. Efforts must always be made to minimize it. The volume of communication (bytes/iteration) is proportional to total length of the interprocessor boundary. With the second constraint in the domain decomposition, the communication volume is minimized.

Fig. 15.1 shows a 16-way partitioning of the Gulf of Maine/Georges Bank model domain. Each color represents the subdomain assigned to a given processor. Note that it is not the geographical area but rather the number of elements that is equal in each subdomain. Partitioning is performed in the horizontal only. The implicit nature of the discretization of the vertical diffusion terms in the FVCOM model make a domain decomposition in the vertical impractical. The partitioning performed by METIS occurs at the beginning of the calculation and requires a trivial amount of time to complete. Statistics on the partitions, including load balance and the number of elements assigned to each processor are written to standard output for reference.
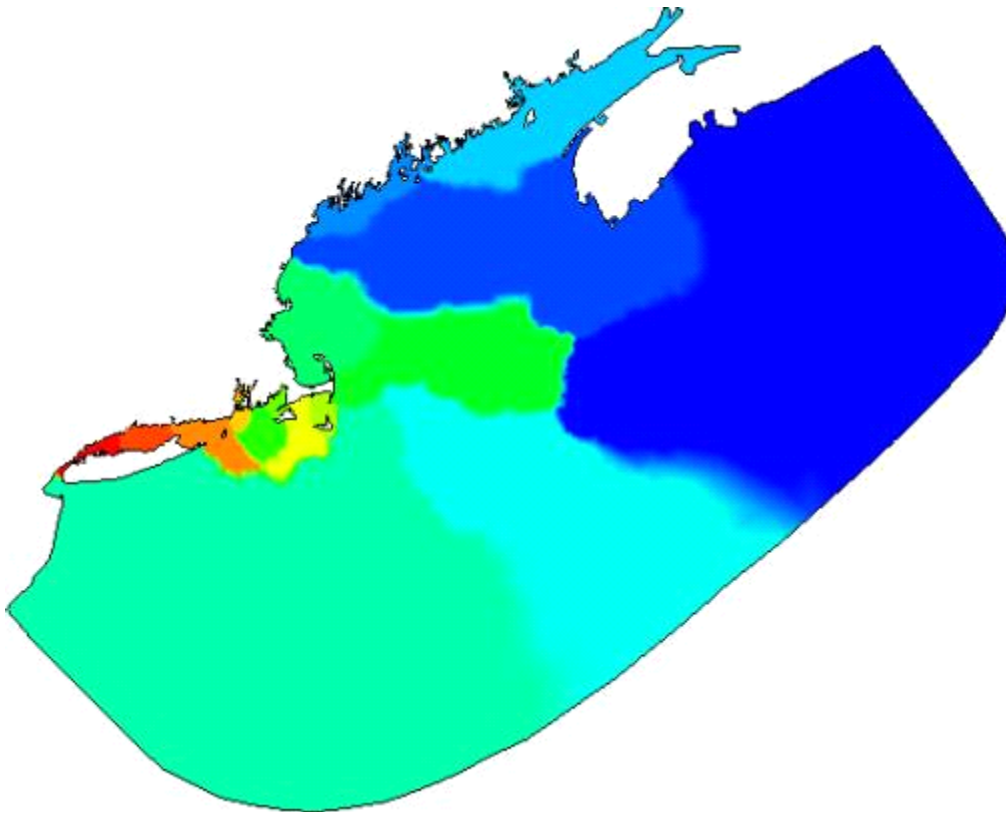
Fig15.1: 16-way partitioning of the Georges Bank/Gulf of Maine domain.

15.2. Domain Setup

After the domain decomposition stage, each processor has been assigned a subdomain in which to integrate FVCOM. Array mappings can then be calculated to map global indices to local and vice versa. Other maps are created to coordinate data exchange across the interprocessor boundaries. Globally referenced data such as rivers, open boundary nodes, and the grid file are decomposed into the local domains using the global to local mappings. For example, if a river runs into a given processor's subdomain, this processor will read in the river inflow data and assign it to the locally numbered node number which corresponds to the inflow river point. The other processors will ignore the data. Open boundaries and spatially varying surface forcing are treated in a similar manner. At the conclusion of this setup stage, each processor has the correct initial and boundary conditions with which to drive a full integration of its subdomain.

15.3. Data Exchange

At the boundaries between processor subdomains, data must be exchanged to preserve the correctness of the flux. The data to be exchanged is set up in a mapping procedure where interior nodes of neighboring processors along the interprocessor boundaries are mapped to the corresponding halo nodes of the exchange partner and vice-versa. For example, in Fig. 15.2, computation of the flux of element E for the edge residing on the interprocessor (dark line) boundary requires information in elements H.
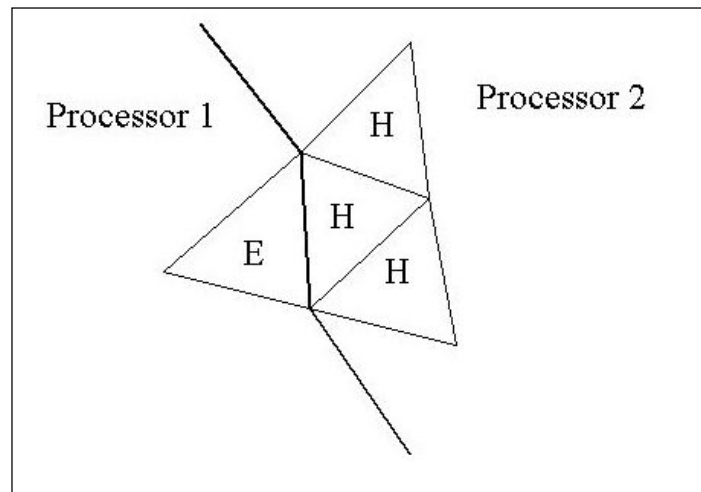
Figure 15.2: Illustration of the data exchange between two decomposition domains.

Elements H belong to Processor 2 (P2) but they are considered halo nodes of Processor 1 (P1). Information on the current state of flow quantities in these elements must be

provided by P2 in order for P1 to compute and update the fluxes of E correctly. This information is provided in an explicit interprocessor communication. Interprocessor communication of data is made using standard MPI (Message Passing Interface) non-blocking send/receive calls. The exchange subroutine is generic and can be used to exchange both element-based and node-based data for both vertically-averaged and three-dimensional flow quantities.

15.4. Data Collection

Flow field output must be performed globally. Thus the data residing in the local processor subdomains must be collected to an aggregate array corresponding to the global element/node numbering defined by the grid file. In general the true values of this array are known only to the master processor that writes the data and discards the array before proceeding with the calculation. Data collection is performed using blocking MPI send/receive pairs.
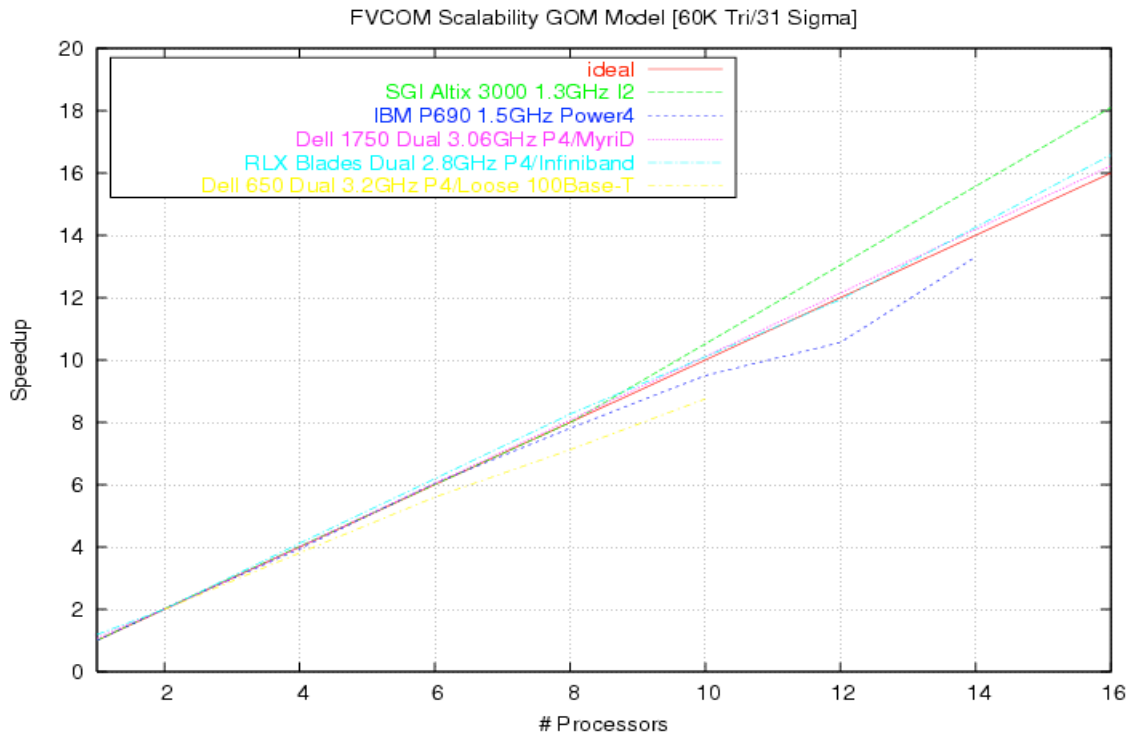
Figure 15.3: The speedup of FVCOM on machines with small number of processors (< 16).

15.5. Performance

The efficiency of the parallel implementation can be analyzed by evaluating the performance speedup. A given model run is performed on an increasing number of processors and the time to complete each run is recorded. Speedup (S) is defined as the
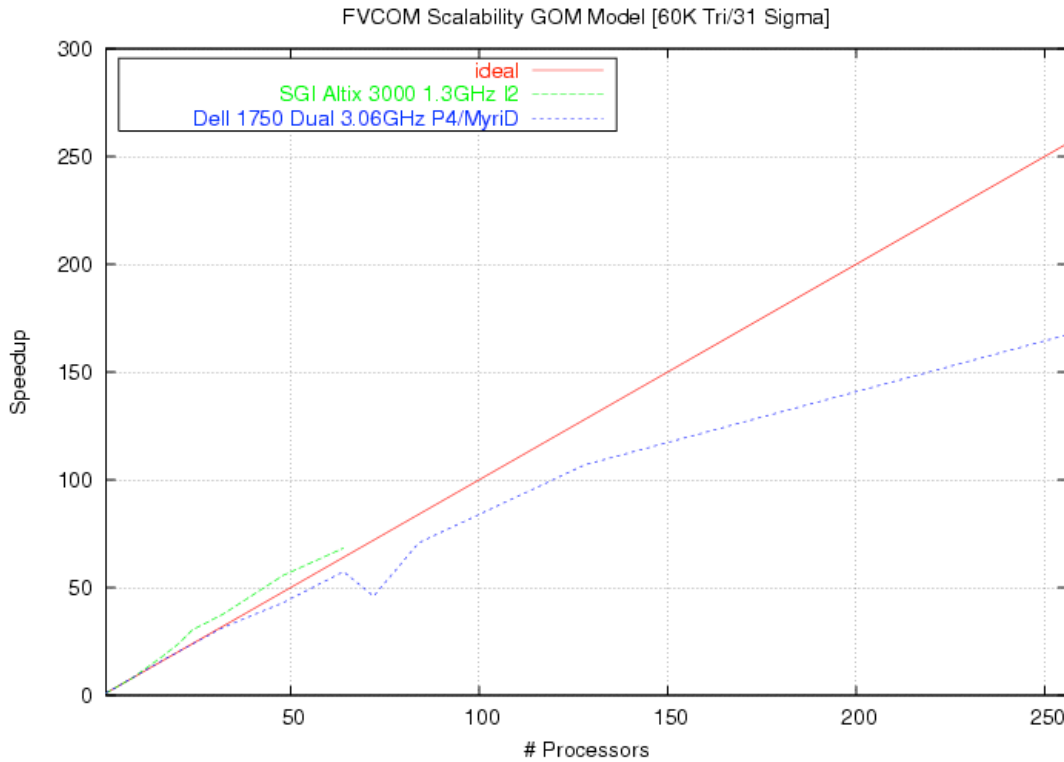
Figure 15.4: The speedup of FVCOM on machines with large number of processors.

ratio of time to complete a job on one processor (serial) divided by the time to complete the job on N processors. Ideally, the curve should be a straight line defined by the equation $S(N) = N$. Various factors combine to modify code efficiency, sometimes resulting in superlinear ($S(N) > N$) speedup. Fig.15.3 shows the speedup measured on systems with small (< 16) number of processors. The curves do not stray far from the ideal line. The SGI Altix exhibits superlinear speedup due to its efficient network, which reduces communication overhead, and the large cache size (3 MB) of the Itanium 2 processor that boosts performance as partition size is decreased. The yellow line corresponds with a loosely coupled cluster made of the desktop computers in the MEDM

lab at SMAST, which are connected by 100BaseT Ethernet. In this case the bandwidth and latency limitations of the interconnect generates significant interprocessor communication overhead, resulting in a significant drop in the parallel efficiency. In the case with large number of processors (Fig. 15.4), the SGI Altix maintains linear speedup up to the maximum number of tested processors (64) and the Dell 1750 series dual Pentium 4 processor nodes coupled with Myricom's Myrinet 2000/D interconnect maintains a speedup of around 100 on 128 processors and 160 on 256, the SGI Altix retains superlinear speedup, but the computational efficiency of the Dell 1750 gradually reduces as more processors are added. It should be noted that speedup is a natural measurement of parallel efficiency but not the primary metric to be evaluated when ranking computers. Job throughput and machine cost are more important criteria when selecting a machine.

A full description of the FVCOM code parallelization, its efficiency, test problems and applications are given in Cowles (2008). It should be noted here that the test was done for FVCOM version 2.4 or lower and no new tests were done for FVCOM version 2.6 or higher. An significant modification was made in FVCOM version 3.1 or higher to improve the model data exchange between nodes. This upgrade helped improve the computational efficiency.

Chapter 16: Model Coding Description and General Information

16.1. What Users Should Know Before Using FVCOM

Many factors can affect the output of numerical circulation models, especially when they are applied to cases with complex bathymetry and realistic forcing. Ideally the model domain and forcing are tailored to the specific scientific application such that the model captures the dominant critical physical processes and the model output supports the understanding of the original scientific problem. Occasionally, there can be some (usually very subtle) mismatch between the model setup and the actual application that produces model results that may agree with some observations but actually contain significant kinematic and/or dynamical errors.   As modelers, we appreciate the difficulties involved in making optimal use of the models. Thus we mention here some important guidelines that should be considered when setting up and running the model.

Users should be familiar with the inherent assumptions in FVCOM. Like most ocean models, the current version of FVCOM is formulated using either the hydrostatic or non-hydrostatic approximation. For example, in many coastal and regional oceanic problems, the hydrostatic approximation works well, but when the study is aimed at simulating very small-scale convection or highly nonlinear internal gravity wave dynamics, the non-hydrostatic approximation is required. In a hydrostatic model run, in order to model vertical convection driven by surface cooling, FVCOM includes an adjustment option to ensure that the density profile is locally statically stable after each time step. However, the true dynamics of vertical convection are driven by non-hydrostatic processes that are not resolved in the hydrostatic model. If the non-hydrostatic FVCOM is used, this adjustment option must be removed. The current version of FVCOM includes the wave-current-sediment interaction. The results of such an interaction depend on choices of the bottom boundary parameterization. We have included a set of parameterizations coded by John Warner at the USGS Woods Hole Science Center. The selection of parameterization heavily depends on users' experiences. FVCOM includes the shortwave irradiance in the water column. The default setup of parameters for the attenuation lengths for longer and shorter wavelength shortwaves and percent of the total shortwave flux associated with the longer wavelength component

were based on research over Georges Bank (Chen et al., 2003b), which may not be applicable for other coastal regions. A review of these parameters was described in Chen et al. (2003b) and users should pay attention to this issue of when the parameters were selected.

Users should be familiar with the important spatial scales of their application. The primary advantage of FVCOM over many coastal ocean models is that the domain is discretized using unstructured triangular meshes. To take advantage of the grid flexibility and maximize the potential of FVCOM in a specific application, it is important to estimate *a priori* the regions where increased grid density is needed. The resulting grid resolution should reflect this result. For example, if a user wants to use FVCOM to resolve the density front at the shelf break, he or she should first investigate the observational information about the cross-shelf scale(s) of this front and use this information to determine the grid density along the shelf. The cross-frontal numerical grid size should be at least 3-4 times smaller than the cross-shelf scale of the front. Users should also understand that FVCOM is a second-order approximation model. This means that the velocity in a MCE is assumed to be linearly distributed, thus setting a limit on the small-scale variation that can be resolved with a given grid resolution For example, if one wants to resolve eddies produced by small-scale topographic features (e.g., canyons) near the shelf-break, the grid in this eddy area must have the resolution to accurately capture the size and current patterns of these eddies.

Users should try to use the right approach to running the model. A good modeler always tries to find out the reason(s) why the model they are using works or fails rather than make a quick conclusion. As more people use FVCOM for different applications, a wider variety of model solutions are generated and the potential for poor model performance and/or poor model/data comparisons increases. In our experience, in many cases, the problems arise due to an inappropriate or incorrect setup of model parameters or the application of the model to a situation beyond the limitations of the model physics. For example, the tides in an estuary are controlled by local bathymetry. Therefore inaccurate results for tidal simulation can be caused by inaccurate bathymetry used for the model configuration. Instead of tuning the model to have a good fit with observations, users should try to improve bathymetric data. Here is our suggestion: if

FVCOM fails to run or produces results counter to physical intuition, please first question yourself and examine carefully the runtime output from the code for setup errors. There is a lot of information provided in this output which is useful for finding setup errors. Second, examine if the model physics can resolve the critical physical processes inherent in your application. Third, ensure that the model temporal and spatial resolutions are sufficient for your problem application. Finding an error in numerical model experiments can require considerable skill and experience, so that we hope that users who do find errors can share their knowledge with other users. Towards this end, we have created a users' forum on the FVCOM website to facilitate communication among users.

FVCOM is in the process of continuous development. The FVCOM code is a complex suite of scientific software. Great efforts have been made to eliminate bugs from the coding. However, we cannot guarantee that it is error free. Users are strongly urged to report bugs to us through the FVCOM Forum or send the development team what they found. Correct findings will be credited to the discoverer. Open issues in model development and performance can be discussed within the community by initiating threads in the FVCOM bulletin board. FVCOM is intended to be a community model and thus feedback and support from the community are critical for continued model development and improvement.

16.2 The Code Structure of FVCOM

The original version of FVCOM (FVCOM 1.0) was written in Fortran 77 and was subsequently migrated to Fortran 90. The Fortran 90 version is organized in a modular fashion, which makes the coding structure clearer and provides the flexibility for users to add and remove modules according to their own research needs. In addition, all arrays are dynamically allocated at runtime so that a change in problem dimensions does not warrant a need for recompilation. Precision is selected using the *selected_real_kind* Fortran intrinsic function. The default precision is single. If a particular application proves sensitive to roundoff errors, the precision can be increased to "double"8 bytes using an option in the code makefile discussed in section 14.2b. Depending on the computer architecture and machine hardware, a change to double precision may increase the runtime by as much as 20-70%. The memory requirement will approximately double.

The first community release version is 2.4 and then up to 2.7. A significant effort has been made to standardize the input and output of FVCOM using NetCDF and the data exchange between nodes. The present version of FVCOM (named FVCOM 3.1.6 or up) system is a fully coupled current-ice-wave-sediment-ecosystem model with significant different code structures from previous versions. FVCOM 3.1.6 or up is solved by either mode-split or semi-implicit time integration schemes. The model can be run in either Cartesian or spherical coordinate systems. We also include a library to use the General
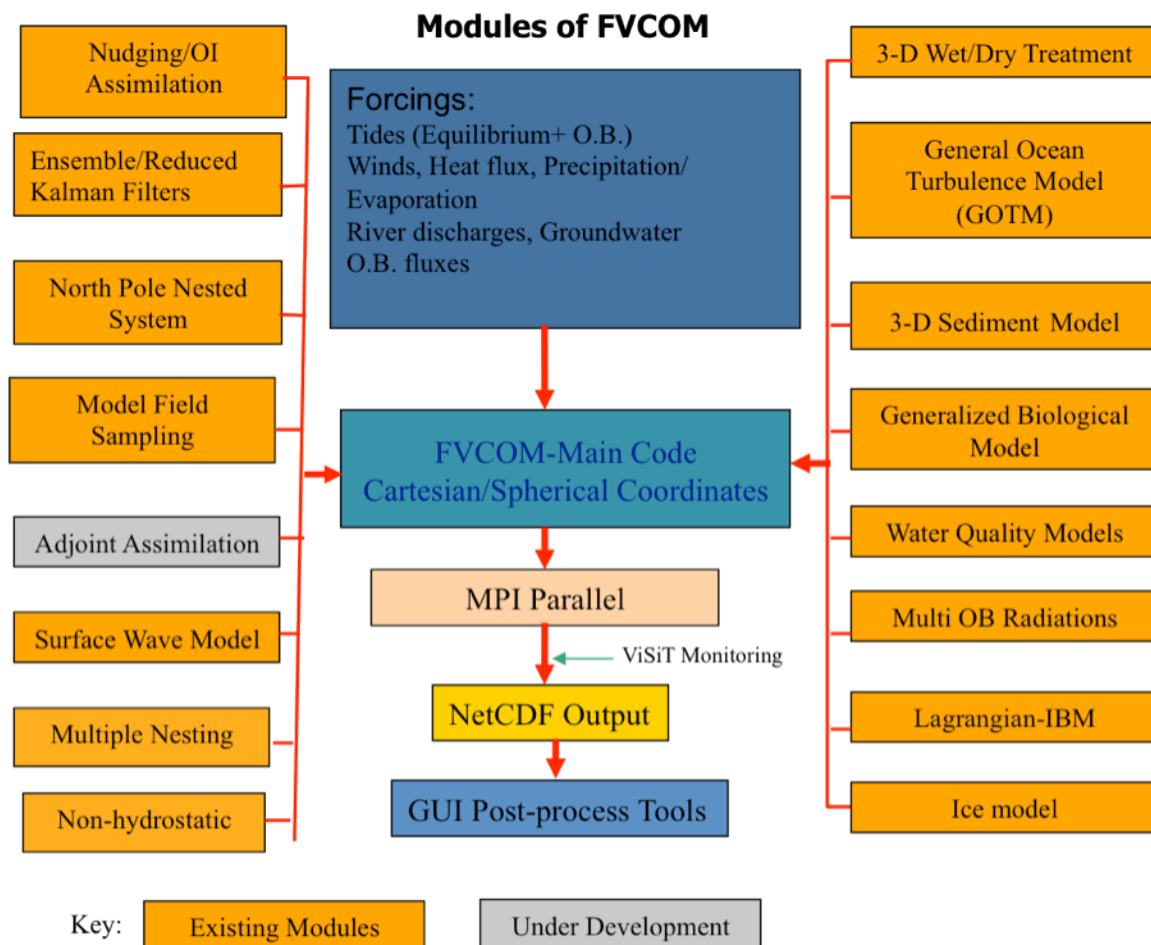
Fig. 16.1: Schematic of FVCOM 3.1.6. The code includes both mode-split and semi-implicit solvers.

Ocean Turbulent Model (GOTM) modules (Burchard et al., 1999; Burchard, 2002) for optional vertical turbulent mixing schemes,

The modules of FVCOM 3.1.6 are listed in Figure 16.1. Brief descriptions of the new modules are given next.

UG-CICE: An unstructured-grid version of the Los Alamos sea ice model Community Ice CodE (CICE). CICE (Hunke and Lipscomb, 2006) is a community ice model that is widely used for polar research. We applied the FVCOM algorithm to convert the structured-grid CICE to the unstructured-grid model (Gao et al, 2011).

FVCOM-NH: a non-hydrostatic version of FVCOM (Lai, 2009). It is coded with both projection and pressure correction methods and has been validated for both idealized and realistic surface and internal solitary waves, lock-exchange flows, and internal waves over steep bottom topography (Lai et al. 2010a-c).

FVCOM-SWAVE: An unstructured-grid version of the Simulating Wave Nearshore model (SWAN) (Qi et al., 2009). SWAN was developed originally by Booij et al. (1999) and improved by the SWAN Team (2006).

FVCOM-SED: An unstructured-grid version of the USGS structured-grid community sediment transport model developed by Warner et al. (2008). With Warner's support, we converted it into FVCOM-SED (Chen et al., 2006d).

FVCOM-BEM: A generalized ecosystem module coupled with FVCOM, which allow users to select either a pre-built biological model (such as a NPZ, NPZD, NPZDB, etc) or construct their own biological model using the pre-defined pool of biological variables and parameterization functions. All these models can be run either coupled with FVCOM (online) or independently by FVCOM output (offline).

FVCOM-WQM: An unstructured-grid finite-volume water quality model based on the EPA Water quality Analysis Simulation Program (WASP).

UG-CE-QUAL-ICM: An unstructured-grid finite-volume version of CE-QUAL-ICM. The CE-QUAL-ICM is the Army Corps of Engineers structured-grid water quality model.

UG-RCA: An unstructured-grid version of the water quality model (RCA) originally developed by (*HydroQual*, 2000, 2004). UG-RCA features 26 water quality state variables and is coded within the FVCOM framework (Chen et al., 2008).

FVCOM also has the following functions: a) 3-D wet/dry point treatment, which can simulate flooding/draining processes in estuaries and wetlands; b) 4-D nudging, OI and Kalman Filters for data assimilation (Chen et al., 2009b); c) the mass conservative nesting module to integrate multi-domain FVCOM domains; and d) the MPI parallelized

visualization tool ViSiT, which allows users to monitor model performance during the simulation and post-process the model output data. The unstructured-grid approach used in the spherical coordinate FVCOM allows this model to run on the global scale
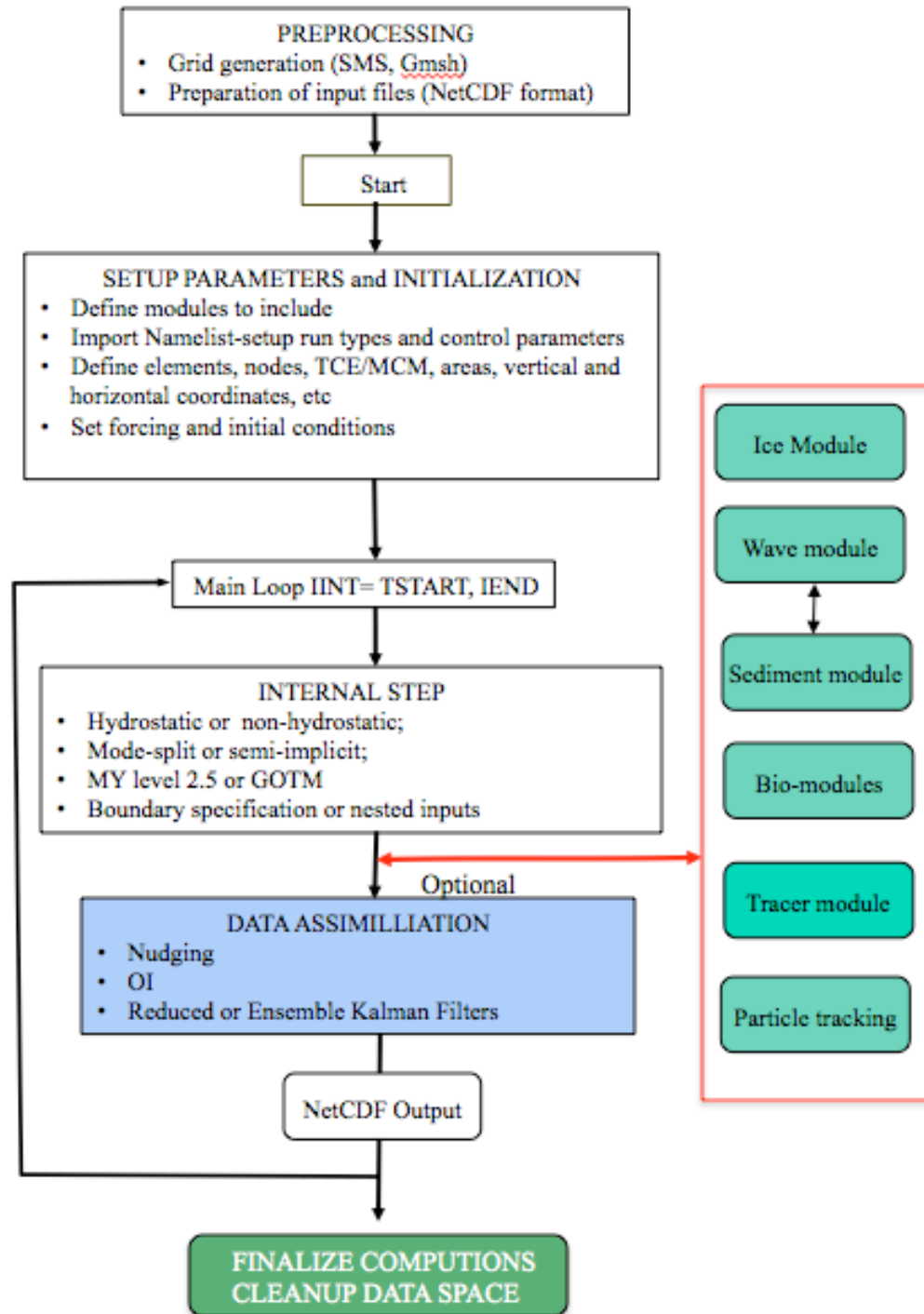
Fig. 16.2: Basic FVCOM flow chart.

including the North Pole.

FVCOM consists of a main control program called "FVCOM.F" (an internal name used in the program is called "USG-FVCOM") and a set of subroutines. A schematic of the flow chart of the baseline code structure is shown in Fig. 16.2. Baseline in this case refers to the flow chart of the core FVCOM code with modules of water surface waves, sediments, water quality, lower trophic food web, tracer-tracking, and data assimilation, etc. The FVCOM code can be run on single processor or multiple-processor Linux or Unix, PC and Mac personal computers or super computers or clusters. The type of the model run is controlled in the makefile. The SMS software is recommended for the unstructured grid creation for FVCOM. However, users may use any two-dimensional unstructured-grid generation program to create unstructured grids used in FVCOM.

16.3 Criterion for Numerical Stability

FVCOM includes mode-split and semi-implicit solvers. For the mode-split solver, the time step used in the external mode of FVCOM is bounded through the Courant-Friedrich Levy (CFL) stability criterion in the form of

$$\Delta t_E \leq \frac{\Delta L}{\sqrt{gD}} \tag{16.1}$$

where Δt_E is the time step of the external mode, the computational length scale ΔL is the shortest edge of an individual triangular grid element, and D is the local depth. This criterion is derived using a linearized surface gravity wave equation, in which diffusion and nonlinear advection are ignored. In most cases it produces a reasonable bound for the true nonlinear case. The time step of the internal mode is restricted by

$$\Delta t_I \leq \frac{\Delta L}{C_I} \tag{16.2}$$

where C_I is the maximum phase speed of internal gravity waves. Since C_I is usually smaller than $C_E = \sqrt{gD}$, Δt_I could be much larger than Δt_E. For normal applications, we usually recommend

$$I_{split} = \frac{\Delta t_I}{\Delta t_E} \leq 10. \tag{16.3}$$

A larger I_{split} could be used in realistic applications, but should be fully tested to check the numerical stability and mass conservation before it is chosen.

In the semi-implicit solver, the time step is restricted by the criterion given as

$$\Delta t_I \leq \frac{\Delta L}{|u|} \tag{16.4}$$

where u is the magnitude of the water current. This criterion is derived based on linear theory, which could differ in the case with strong nonlinearity. We found that when the grid size becomes very small in an estuary or coastal region, the time step could be the same as that used for the internal model in the mode-split solver. However, when the grid size is bigger, the time step could be much larger.

16.4 Subroutine and Function Descriptions

The general functionality of the major subroutines used in the FVCOM core code is listed below. Users should be aware that certain subroutines in modules developed for high commercial use may not be available in the publically released version. For users who want to use those modules, please contact Dr. Chen to get permission. We require information on the objectives of projects before we can provide those modules.

adcor.F	Correct the Coriolis term using a semi-implicitly code to avoid inertial instability.
adjust_ts.F	Control the mass conservation of temperature and salinity for the horizontal advection calculation at river mouths (for the case with no selection of MPDATA)
adjust2d3d.F	Adjust the internal mode velocity to the external mode velocity. This is done by using the defect between updated and current vertically averaged velocities.
adv_q.F	Calculate the terms of turbulent advection, shear and buoyancy production, dissipation, and horizontal diffusion in the Mellor and Yamada level 2.5 turbulent kinetic energy (q^2) and turbulent macroscale-related (q^2l) equations.
adv_s.F	Calculate the terms of advection and horizontal diffusion in the salinity equation.

adv_t.F	Calculate the terms of advection and horizontal diffusion in the temperature equation.
adv_uv_edge_gcn.F	Calculate the terms of advection, Coriolis, pressure gradient and horizontal diffusion in the x and y momentum equations for selection with no ghost cell boundary treatment.
adv_uv_edge_gcy.F	Calculate the terms of advection, Coriolis, pressure gradient and horizontal diffusion in the x and y momentum equations for selection with ghost cell boundary treatment.
advave_edge_gcn.F	Calculate the fluxes of advection and diffusion in the external mode momentum equations for selection with no ghost cell boundary treatment.
advave_edge_gcy.F	Calculate the fluxes of advection and diffusion in the external mode momentum equations for selection with ghost cell boundary treatment.
advection_edge_gcn.F	Calculate the terms of 3-D advection and horizontal diffusion that are used to compute the vertically-averaged advection and diffusion terms related to the 3-D flow field in external mode momentum equations (Gx and Gy) for selection with no ghost cell boundary treatment.
advection_edge_gcy.F	Calculate the terms of 3-D advection and horizontal diffusion that are used to compute the vertically-averaged advection and diffusion terms related to the 3-D flow field in external mode momentum equations (Gx and Gy) for selection with ghost cell boundary treatment.
allocate_all.F	Allocate and initialize arrays.
baropg.F	Calculate the horizontal baroclinic pressure gradient in a selected generalized terrain-following coordinate.
bcond_gcn.F	Set boundary conditions: tidal forcing at open boundaries, no-flux of external and internal mode currents on the solid wall; surface forcing for external and internal modes. GCN represents the selection with no ghost cell treatment.

bcond_gcy.F	Set boundary conditions: tidal forcing at open boundaries, no-flux of external and internal mode currents on the solid wall; surface forcing for external and internal modes. GCY represents the selection with ghost cell treatment.
bcond_ts.F	Set up the open boundary conditions for temperature and salinity.
brough.F	Calculate the bottom drag coefficient.
calc_vort.F	Calculate the vorticty.
cell_area.F	Calculate the area of individual triangles as well as areas for node- and element-based flux control volumes.
cntrl_prmtrs.F	Namelist control parameters.
CLOSEFILE.F	Close the output files.
conv_over.F	Adjust for static instability in the water column.
coords_n_const.F	Load/setup physical quantities (Coriolis, gravity, sponge_layer, X and Y or longitude and latitude)
depth_check.F	Check the water depth to be sure it is greater than minimum depth in the case with no wet/dry point treatment. When the water depth is less than the minimum depth, halt the program with a warning message on the screen.
depth_grad.F	Calculate the spatial bathymetric gradients.
edge_len.F	Calculate the length between an edge point to the node or centroid for TCE and MCE, respectively.
enkf_ncdio.F	Read ensemble model-run results from the NetCDF files and writing the EnKF analysis results back to this NetCDF file.
eqs_of_state.F	Calculate the water density with three different methods: DENS1, DENS2, and DENS3.

DENS1: calculate the potential density using potential temperature and salinity following Fofonoff &Millard's method. The pressure effects are |

	incorporated to allow the model to run for the freshwater case with water temperature < 4 ^{o}C. Users can set up PR = 0 for a default value. That means that the potential density is calculated with a reference at the sea surface. If the model domain includes the deep ocean, specifying PR=0 can produce a larger value of density in the deeper water, which could lead to a significant topographic coordinate error. DENS2: calculate the in-situ density (sigma-t) based on salinity and temperature. This method is the same as that used in POM. For the estuaries and inner shelf, DENS1 is sufficient. DENS3: calculate the in-situ density based on a polynomial expression (Jackett and McDougall, 1995) in which the pressure is taken into account. This method can be used for both shallow and deep waters.
extel_edge.F	Calculate the advective fluxes of the vertically integrated continuity equation.
extelpf_edge.F	Calculate the subtidal surface elevation at time step n+1.
external_step.F	External integration subroutine, which is called in "Internal_step.F".
extuv_edge.F	Accumulate the fluxes of the external modes.
fct_q2.F	Correct the vertical advection term calculation using a flux corrected transport scheme for the q2 equation.
fct_q2l.F	Correct the vertical advection term calculation using a flux corrected transport scheme for the q2l equation.
fct_s.F	Correct the vertical advection term calculation using the Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) for the salinity equation.
fct_t.F	Correct the vertical advection term calculation using the Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) for the temperature equation.

fvcom.F	The main program of the FVCOM system.
genmap_lsf. F	Set up global-local node and element numbering maps and inter-processor communication for the case with user's specification of the upstream cross-shelf boundary for wind-induced flux. This is used for the regional Gulf of Maine FVCOM run for the case with no nesting boundary condition is included. It should not be turn on for a general use of FVCOM for the other region.
genmap_obc.F	Set up Set up global-local node and element numbering maps and inter-processor communication for the open boundary cells..
genmap.F	Set up global-local node and element numbering maps and inter-processor communication.
ghostuv.F	Calculate ghost cell velocity for external mode and internal mode.
grid_metrics.F	Unstructured grid modules in which the grid is defined, flux and control volumes, element and control volume areas, shape coefficients, edge length, shape factor, bathymetry gradient, boundary meshes, etc are defined or calculated.
ice_albedo.F	Determine the snow and ice albedo parameterizations. This subroutine was written by Dr. Elizabeth C. Hunk at LANL for CICE.
ice_atmo.F	Ice-atmospheric interface module in which the physical variables in the atmospheric boundary layers are determined and air-ice interface stability, flux, stress, etc are calculated. This is a subroutine of CICE.
ice_calendar.F	Calendar routine for managing the time used in the ice model. This is a subroutine of CICE.
ice_constants.F	Set up physical constants used in the ice model. This is a subroutine of CICE.
ice_coupling.F	The module to set up message passing to and from the ice-ocean model coupler. This subroutine was

	originally set up for CICE and modified to couple FVCOM.
ice_domain.F	Set up array size for local subdomain and multi-processor communication. The code was originally from the "domain.F in POP and modified to meet the FVCOM environment.
ice_fileunits.F	Define unit numbers for files opened for reading or writing. This is a subroutine of CICE.
ice_flux_in.F	Read and interpolate input forcing data in the ice model. This is a subroutine of CICE and modified to meet the FVCOM environment.
ice_flux.F	Flux variable declarations: coupler, diagnostic and internal. Flux variable declaration includes fields sent from the coupler (defined as "in"), sent to the coupler (defined as "out"), written to diagnostic history files (defined as "diagnostic") and used internally (defined as "internal"). This is a subroutine of CICE.
ice_grid.F	Set up spatial grids, masks and boundary conditions for the ice model. This subroutine was originally written for CICE and modified for the use in the unstructured grid in FVCOM.
ice_init.F	The ice model parameters and variables' initialization.
ice_itd_linear.F	Linear remapping scheme for the ice thickness distribution.
ice_itd.F	Initialize the ice thickness distribution and utilities to redistribute the ice among categories. This is a subroutine of CICE.
ice_kinds_mod.F	Define variable precision for all common data types. This is a subroutine of CICE.
ice_mechred.F	Ice mechanical redistribution (ridging) and strength computations. This is a subroutine of CICE.
ice_model_size.F	Define the global domain size and number of categories and layers in the ice model. This is a subroutine of CICE.

ice_ocean.F	Calculate the ocean mixed layer (internal to the sea ice model). This subroutine also allows heat storage in ocean for ice-ocean uncoupled runs.
ice_scaling.F	Scale ice fluxes by the ice area.
ice_state.F	Define and set up primary state variables in various configurations.
ice_therm_itd.F	Thermodynamics calculations after call to coupler (the ice thickness redistribution, lateral growth and melting and freeboard adjustment).
ice_therm_vertical.F	Update ice and snow internal temperature and compute thermodynamic growth rates and atmospheric fluxes.
ice_work.F	Globally accessible, temporary work arrays. These arrays are used to save memory by allocating global arrays only when necessary.
icing.F	This is a subroutine to calculate the icing rate at the sea surface using FVCOM output.
init_sed.F	FVCOM sediment module initialization routine (user defined).
internal_step.F	Internal time step integration module. This module also includes the external time step integration module.
linklist.F	Link list module that create link, insert or delete nodes in the list.
load_grid.F	Get the grid connectivity and open boundary condition node list for domain decomposition and genmap.
longshore_flow.F	Set up the metrics for longshore wind driven flow. This subroutine was created for the regional Gulf of Maine FVCOM run without inclusion of nesting boundary. For a general use of FVCOM, it should not be turned on.
mod_action_ex.F	Explicit scheme solver module for the wave action density equation.

mod_action_im.F	Implicit scheme solver module for the wave action density equation.
mod_balance_2d.F	Output terms in 2-D momentum equations for the momentum balance analysis.
mod_bbl.F	The bottom boundary layer module with wave-current interactions. This module was originally written by Dr. J. Warner (USGS) for ROMs, and modified to meet the FVCOM environment.
mod_bio_3D.F	This is a module to convert a 1-D biological module to the 3-D biological module under the FVCOM framework.
mod_boundschk.F	Module used to check if variables are in user-defined bounds
mod_bulk.F	Module used to calculate the wind stress
mod_clock.F	Timing module:
mod_dam.F	Dike-groyne module used to include the right fluid dynamics across a vertical wall under or above the sea surface.
mod_dye.F	The dye-tracking module.
mod_enkf.F	Setup the EnkF name list and functions for initial configuration.
mod_enkf_ncd.F	Read and write the dimensions (size and dimension names), static variables and real time dynamics variables used for EnKF module.
mod_enkf_obs.F	Read observational data with the same format as nudging/OI assimilation input file and then convert them into the format used for EnKF.
mod_enkfassim.F	The main subroutine for ensemble-based KFs assimilation (including EnKF, ESRF, LETKF, ETKF, and SIEK).
mod_force.F	Module used to setup and update forcing variables such as the surface wind, surface heat flux, air

	pressure, precipitation, river discharges, groundwater fluxes, ice forcing, and tides, temperature and salinity on the open boundary, etc.
mod_gotm.F	Module used to link GOTM libraries
mod_heatflux.F	Module used to recalculate the heat flux.
mod_ice.F	Module to couple the ocean and ice models. It includes subroutines for ice variable definition and initialization as well as a coupler for FVCOM and UG-CICE.
mod_ice2d.F	Module used to solve the ice momentum equations.
mod_input.F	Module for inputs, which is used to initialize, read and print namelist; open files and load messages of grid, depth, Coriolis and vertical layers.
mod_interp.F	Interpolation module, which includes several interpolation subroutines.
mod_lag.F	Lagrange particle tracking module.
mod_main_wave.F	Wave module that defines variables and arrays in SWAVE.
mod_main.F	Module in which global limits and array sizing parameters, control variables, parameters in the namelist are defined.
mod_meanflow.F	Module used to specify the mean flow on open boundary.
mod_ncdio.F	Set up the NetCDF format output.
mod_ncll.F	Cross-link list NetCDF file module.
mod_nctools.F	NetCDF tool library module.
mod_nesting.F	Nesting module used to nest a large domain to a small domain. It includes the large domain output and the small domain input.
mod_newinp.F	OSCAR input utility module

mod_non_hydro.F	Nonhydrostatic dynamics module
mod_northpole.F	North pole treatment module.
mod_obcs.F	Module used to specify and calculate the open boundary conditions of elevation, temperature and salinity.
mod_obcs2.F	Module used to specify the open boundary condition of the mean flow.
mod_obcs3.F	HCQ open boundary flux module
mod_onedtide.F	Equilibrium tidal module
mod_optimal_interpolation.F	Module for n-dimensional optimal interpolation
mod_par.F	Parallelization module
mod_petcs.F	Interface to use PETs sparse matrix software in FVCOM
mod_plbc.F	Periodic lateral boundary condition module
mod_prec.F	Definite floating point precision using selected real kind intrinsic function.
mod_probe.F	Set up time series printing.
mod_pwp.F	Calculate the surface mixed layer depth based on the PWP model.
mod_report.F	Report module for the flow field statistics during runtime monitoring of solution.
mod_rrk.F	The main subroutine for the Reduced Rank Kalman Filter (RRKF) operation.
mod_rrkassim.F	An old RRKF subroutine used in FVCOM version 2.7 and was removed in the FVCOM version 3.1 or up.
mod_scal.F	FVCOM scalar module used for sediment model.
mod_sed.F	Sediment model module

mod_semi_implicit.F	Free surface elevation solver module using a semi-implicit time stepping method.
mod_set_time.F	Model time and time step setup module
mod_setup.F	Set up module for grid, depth, Coriolis, horizontal mixing coefficient, bottom roughness and vertical layers et al.
mod_sng.F	Library of Fortran string manipulation routines
mod_spherical.F	Spherical coordinate module
mod_startup.F	Startup module that contains different types of the model startup.
mod_station_timeseries.F	Time series output module for selected stations.
mod_time.F	Time treatment module
mod_tridiag.F	Tri-diagonal matrices' solver module.
mod_types.F	User-defined type module
mod_utils.F	Utility module
mod_visit.F	ViSIT-based FVCOM monitoring interface module.
mod_wave_current_interaction.F	Wave-current interaction module.
mod_wavesetup.F	Wave setup module. This module was created when the SWAVE was developed. After SWAVE is coupled with FVCOM, this module was not used anymore.
mod_wd.F	Wet/dry point treatment module
mod_wqm.F	Define parameters and arrays for the water quality module: 1) Dissolved oxygen (DO) 2) Carbonaceous biochemical oxygen demand (CBOD) 3) Phytoplankton (PHYT) 4) Ammonia nitrogen (NH4) 5) Nitrate and Nitrite Nitrogen (NO3+NO2) 6) Organic Nitrogen (NH4)

	7) Orthophosphorus or Inorganic phosphorus (OPO4) 8) Organic phosphorus (OP)
namelist.F	Namelists for model setup are defined in "Mod_main.F" and the programs for namelists' initialization, reading and printing are in "Mod_input.F". This subroutine is to make the use of the subroutines defined in those two modules
nh_set_nesting.F	Setup the subroutine nesting when nonhydrographic dynamics turn on.
ocpcre.F	Ocean pack command reading routines used for SWAN, which include subroutines used for reading.
ocpids.F	Ocean pack installation dependent routines used in SWAN.
ocpmix.F	Ocean pack miscellaneous routines used in SWAN.
open_all.F	Open most of input and output files that are not included in "Mod_input.F".
particle.F	Lagrangian particle tracking module.
phy_baropg.F	Calculate the baroclinic pressure gradient in the standard z-levels. The water column is divided into 600 standard levels and pressure gradient is calculated at each level. The resulting pressure gradients are converted back to sigma-levels through vertical interpolation approach. (The test case shows that the small errors near the bottom on steep bottom topography due to this calculation might grow through nonlinear interaction. Recommend to use only for comparison between the z and sigma coordinate systems, but not for realistic applications.)
print_vals.F	Print variables in paralleling mode.
rho_pmean.F	Calculate the mean of density at standard z levels.
sectinf.F	Specify the output files used for the "GRI" graphics configured by the Marine Ecosystem Dynamics Laboratory at SMAST/UMASSD.
setup_domain.F	Using METIS to define local domains and create

	global to local and local to global maps
Shape_coef_gcn.F	Calculate the coefficient for a linear interpolation function on the x-y plane for the non-ghost cell case.
shape_coef_gcy.F	Calculate the coefficient for a linear interpolation function on the x-y plane for the ghost cell case.
sinter.F	An array interpolation and extrapolation subroutine
startup_type.F	Startup types setup
swancom1.F	Simulate the wave energy in SWAN. This includes a subroutine "SWCOMP", which carries out the main processes that are determined in several other subroutines.
swancom2.F	Calculate the source terms of bottom friction, wave breaking, white capping, etc in SWAN.
swancom3.F	Calculate the source terms of wind input in SWAN.
swancom4.F	Calculate the source terms of wave-wave interaction in SWAN.
swancom5.F	Compute the wave number, the group velocity, the propagation velocities of the energy in X, Y, S and D spaces.
swanmain.F	Main subroutine of the SWAN wave model, which includes subroutines for initializing and pre-processes before the time integration starts.
swanpre1.F	Read and process the user commands described in the SWAN model.
swanpre2.F	Read and process the boundary commands.
swanser.F	SWAN service routines.
swmod1.F	Common variables related modules of the wave model: file 1 of 3.
swmod2.F	Common variables related modules of the wave model: file 2 of 3.

swmod3.F	Common variables related modules of wave model: file 3 of 3.
tge.F	Define the non-overlapped, unstructured triangular meshes used for flux computations. The mesh could be created using the commercial software called "sms8.0" or other mesh generation programs. The mesh file generated by sms8.0 can be used directly with this subroutine, while the mesh file generated using other programs must be converted to meet the required format used here.
vdif_q.F	Calculate implicitly the vertical diffusion terms of turbulent kinetic energy and macroscale equations.
vdif_ts_gom.F	Same as file vdif_ts.F, but only used for Gulf of Maine model. In general use, this program should not be included.
vdif_ts.F	Calculate implicitly the vertical diffusion terms of water temperature and salinity equations.
vdif_uv.F	Calculate the vertical diffusion terms of the u and v momentum equations.
vertvl_edge.F	Calculate the topographic coordinate vertical velocity for the 3-D mode.
viscofh.F	Calculate the horizontal diffusion coefficient.
visitsim.F	ViSIT online monitoring program.
wreal.F	Calculate the Cartesian vertical velocity.

Bio-Sources

This subdirectory includes the subroutines and modules for the online general biological models (FVCOM-GEM). With users' specification, a low trophic level food web model can be built using these programs. By changing the definitions of each program, one could easily extend this model to the high trophic level. We have already created an offline version of this model (see the offline modules). These programs listed here are all required for the offline model, too.

bacteria.F	Compute the source and sink terms of bacteria in the 1-D base without inclusion of vertical mixing.
bio_mixing.F	Solve the vertical diffusion terms for biological equations.
detritus.F	Compute the source and sink terms of detritus in the 1-D base without inclusion of vertical mixing
dom.F	Compute the source and sink terms of DOM in the 1-D base without inclusion of vertical mixing.
Get_parameter.F	Reads control parameters for the user-built biological model.
mod_1D.F	Define parameters related to a 1-D application.
mod_bacteria.F	Defines parameters related to bacteria
mod_detritus.F	Defines parameters related to detritus.
mod_dom.F	Defines parameters related to DOM.
mod_nutrient.F	Define parameters related to nutrients.
mod_phytoplankton.F	Defines parameters related to phytoplankton.
mod_zooplankton.F	Defines parameters related to zooplankton.
nutrient.F	Compute the source and sink terms of nutrients in the 1-D base without inclusion of vertical mixing.
phytoplankton.F	Computed the source and sink terms of phytoplankton in the 1-D base without inclusion of vertical mixing.
zooplankton.F	Compute the source and sink terms of zooplankton in the 1-D base without inclusion of vertical mixing.

FVCOM also includes several offline models, which are driven by the physical field output of FVCOM. They include: 1) UG_GEM; 2) UG_RCA; 3) UG_CE-QUAL-ICM and 4) LAG. UG-GEM is the generalized lower trophic food web model, UG_RCA is the unstructured grid version of the water quality model (Row/column version of AESOP), and UG-CE-QUAL-ICM is the unstructured-grid version of an integrated compartment

water quality model (CE-QUAL-ICM). CE-QUAL-ICM was originally developed by C. F. Cerco at the U.S. Army Corps of Engineers, converted to the unstructured-grid version by J. Qi and C. Chen, and modified and validated by T. Kim and R. G. Labiosa at PNNL.

UG-RCA	
allocate_rca.F	Allocates and initialize the RCA arrays. This is a program written for RCA.
allocate_sed.F	Allocates and initialize the sediment arrays in RCA. This is a program written for RCA.
cell_area.F	Calculates the area of individual triangle based on the three vertex coordinates and calculate the topographically coordinate surface area of individual control volume consisted of triangles with a common node point. This subroutine was from the FVCOM source code.
domdec.F	Domain decomposition adopted from the FVCOM source code.
eutro.F	RCAS Eutrophication model subroutines.
fct_s.F	Flux control for overshooting issue in the salinity flux calculation. The salinity is computed in RCA. In the UG-RCA, we have added an option to use the salinity from the physical model. Since the data assimilation is usually done for salinity in the physical model. It is better to use the salinity from the physical model output. If this is selected, the salinity will not calculated in UG-RCA, and all programs used to calculate the salinity will not be used.

Since we have used the salinity from the physical model output, we have not spent the time on checking the salinity calculation program in UG-RCA. The bugs may be in the boundary values (input or output) computation. |
fmter.F	Error reporter.
genmap.F	Mapping cells and nodes. This is the program from the FVCOM source code.
iunitcheck.F	Tiim units' checker.
mod_ncd.F	NETCDF module (see the definition in the FVCOM source code).

mod_par.F	Parallel computation module (see the definition in the FVCOM source code).
mod_prec.F	Definite floating point precision using selected real kind intrinsic function (from the FVCOM source code).
mod_rca.F	Defines RCA variables.
mod_sed.F	Defines sediment variables.
mod_types.F	Defines variable types.
mod_utils.F	IO error reporter.
mod_var.F	Define global limits and array sizing parameters.
ncdio.F	NETCDF reading and writing.
pdomdec.F	Set up local domain for parallelization.
print_vals.F	Local values reporter.
rca.F	Main controlling and operation program for RCA.
rca01.F	RCA initialization.
rca02.F	IO controller.
rca03.F	Read the model geometry and initial transport fields as provided by the hydrodynamics model.
rca04.F	Boundary condition loading.
rca05.F	Read the forcing functions
rca06.F	Read the parameter values.
rca07.F	Read the initial conditions.
rca08.F	Set up synchronize data files.
rca09.F	Output routine for intermediate dump.
rca10.F	Updates forcing functions.

rca11.F	Updates boundary conditions.
rca12.F	Prints user required dumps at the end of simulation.
rca16.F	Updates the misc time functions.
rcaadv.F	The routine used to integrate the differential equations of the water quality model. This is modified to fit FVCOM unstructured grid environment.
rcabyss.F	Produce a system dump file from the master dump file (rac11.F).
rcadwrit.F	Write RCA dumps to disk,
rcamess.F	Reporter of the change in integration and stability during the simulation.
rcamprof.F	Calculate the domain-averaged concentrations of the water quality variables.
rcaprnt.F	General array printing routine.
sedmodlsubs.F	Sediment benthic processes routine.
sinter.F	Interpolation routine.
stoich.F	Calculate the ratio of nutrients to carbon.
tge.F	Definition of triangular grid, which is the same as "Triangle_grid_edge.F".
utilties.F	Utility files for the use of halting program correctly.
vdif.F	Implicit solver for vertical diffusion terms in the UG-RCA.
vertvl.F	Calculate the vertical velocity in the topographic coordinate system.
viscofh.F	Calculate the horizontal advection and diffusion terms in the UG-RCA.
vssedmodl.F	Variable stoichiometry sediment submodel.

UG-GEM	
adjust_bio.F	Adjust bio_variables near river mouths using adjacent nodes.
alloc_vars.F	Allocate and initial arrays.
archive.F	Archive the model results.
bcmap.F	Define and specify the boundary conditions.
bcs_force.F	Set up the boundary conditions: bottom freshwater flux and freshwater river discharges.
bracket.F	Determine data interval and time interpolation.
cell_area.F	Calculates the area of individual triangle and the area of individual control volume in the topographically following coordinate.
closefiles.F	File closers
data_run.F	Read control parameters for the model run.
domdec.F	Domain decomposition.
fct_bio.F	Flux controller.
fvcom_gem.F	Maine program for GEM.
genmap.F	Grid mapping for local node and element numbering.
getbdno.F	Determine the number of nodes on outer boundaries.
iofiles.F	Open IO files.
mod_bio_3D.F	Convert a 1-D biological model to the 3-D version linked with FVCOM.
mod_bio_obc.F	Open boundary condition module.
mod_clock.F	O'clock time computation.
mod_inp.F	Input file module.

mod_main.F	Module in which global limits and array sizing parameters, control variables, parameters in the namelist are defined.
mod_ncdin.F	Set up the NetCDF format input.
mod_ncdout.F	Set up the NetCDF format output.
mod_northpole.F	Define the north pole array in spherical coordinate.
mod_obcs.F	Setup the boundary conditions
mod_onedtide.F	1-D application module.
mod_par.F	Defines parallel variables.
mod_prec.F	Definite floating point precision using selected real kind intrinsic function.
mod_probe.F	Setup the time series output at user's selected stations.
mod_spherical.F	Defines the spherical coordinate.
mod_tsobc.F	Define and determine the temperature and salinity boundary conditions.
mod_types.F	Defines variables types.
mod_utils.F	Error reporter.
ncdio.F	Read in the data from the NetCDF files.
pdomdec.F	Set up local domain for parallelization.
print_vals.F	Local values reporter.
report.F	Report initial information.
sinter.F	Interpolation program.
startup.F	Begin restart run from specified time.
tge.F	Definition of triangular grid, which is the same as "Triangle_grid_edge.F".

utiltties.F	Program stopper
vertvl.F	Compute the vertical velocity in a topographically following coordinate system.
viscofh.F	Compute the advection and horizontal diffusion coefficient by solving the advection and horizontal diffusion terms in the biological model.
water_depth.F	Read in static water depth and calculate related variables.

UG-CE-QUAL-ICM	
adv_wqm.F_	Solve advection and horizontal diffusion terms in the water quality equation.
bcmap.F	Map open boundary nodes to local processors.
bcond_wqm.F	Set boundary conditions of water quality variables.
bcs_force.F	Set up the boundary conditions of water quality input from rivers.
bracket.F	Determine the data interval for the current time and for time dependent boundary forcing.
cell_area.F	Calculate the area of individual triangles and the areas for node- and element-based flux control volumes.
domdec.F	Partition the domain into elements using the METIS graph partitioning libraries.
fct_nut.F	Flux control for nutrients.
genmap.F	Define local-to-global node and element numbering maps. Set up arrays to control inter-processor data exchanges.
hydro.F	Read the velocity field from the NetCDF file.
mod_ncd.F	Function and subroutines of the NetCDF input and output.
mod_obcs.F	Open boundary conditions of water quality variables.

mod_par.F	Parallelization module.
mod_prec.F	Definite the floating point precision using selected real kind intrinsic function.
mod_types.F	User-defined type module.
mod_utils.F	Utility module.
mod_var.F	Contains primary code data for UG-CE-QUAL-ICM model.
ncdio_spechydro.F	Same as Read.F except NOBTY -> NOBTY + 1.
ncdio.F	Read in data from the NetCDF files.
pdomdec.F	Set up local domain for parallelization.
tge.F	Define the non-overlapped, unstructured triangular meshes used for flux computations.
tvds.F	Reading in meteorological, benthic flux and submerged aquatic vegetation data.
utilties.F	Utility file PSTOP.
vdif_wqm.F	Solve implicitly the vertical diffusion terms of the water quality equation.
vertvl.F	Calculate the vertical velocity in the topographically following coordinate system.
viscofh.F	Calculate the horizontal diffusion coefficient in the water quality equations
wqm_alg_biosoln.F	Same as wqm_alg.F.
wqm_alg.F	Algal subroutines.
wqm_inputs_parallelcdf.F	Same as wqm_inputs.F except the path of file is different.
wqm_inputs.F	Read in control and other files and initialize computational variables.
wqm_kin_biosoln.F	Same as wqm_kin.F but DOR1 changed for the analytical solution.

wqm_kin.F	Kinetics subroutines for water quality variables,
wqm_main.F	Main program of UG-CE-QUAL-ICM.
wqm_modules.F	Define water quality variables and allocate the water quality variable arrays.
wqm_owq.F	Compute light attenuation and irradiance.
wqm_sav.F	Submerged aquatic vegetation routine.
wqm_sed.F	Sediment modules for the water quality model.
wqm_sf.F	Ruspension feeder routine used for the water quality model.

Offline Lagrangian Particle Tracking Program	
alloc_vars.f90	Define and allocate variables and arrays used for particle tracking.
data_run.f90	Input control parameters, read in variables and set up values, set up the time steps for time integration and output, specify the option for random walk, and set up unit for the input and output files.
mod_inp.f90	Input file module.
mod_ncd.f90	NETCDF module (see the definition in the FVCOM source code).
mod_prec.f90	Define the floating-point precision.
mod_var.f90	Define global limits and array sizing parameters for the particle-tracking programs.
ncdio.f90	Define NetCDF format input and output.
offlag.f90	Main program for the particle tracking.
triangle_grid_edge.f90	Define the unstructured grid mesh.
util.f90	Utility programs.

Chapter 17: Model Installation, Compilation, and Execution

Acquisition, installation, compilation, and execution of the model are described in this chapter. Note that most of these steps are machine dependent and some knowledge of the given computer system will be necessary. Much work has been done to make the model portable. However, FVCOM is a rapidly evolving research tool, not a release of commercial software, and thus installation and compilation of the model may not be straightforward. Section 17.1 will address downloading and unpacking the model. The model directory tree will be shown. Section 17.2 will discuss model compilation. Section 11.3 will present the normal procedure of model execution.

Windows Users: It is recommended that you download and install the latest version of *cygwin* (http://www.cygwin.com). *Cygwin* is a bash shell/Unix emulation program and contains many of the tools such as *tar*, *gzip/gunzip*, and *cpp* which will be useful for installation and compilation of FVCOM. Also, *cygwin* gives Windows a decent interface with which to setup and run the model. Note that it requires basic familiarity with Unix commands.

17.1. Obtaining FVCOM

FVCOM can be obtained from the FVCOM community website through the code repository portal. Note that this section of the website is password protected. Usernames and passwords will be provided after the user agreement form has been signed. Once in the repository, read the package definitions and choose the appropriate version for your research needs. Download the gzipped tar file directly to your computer. This tar file will contain the source code, a makefile, and the test problems described in Chapter 15. Place the model in a suitable location and unpack with:

<div align="center">gunzip –c FVCOMxxx | tar xvf -</div>

This will produce the top level directory FVCOMxxx. The directory structure is shown in Fig. 17.1.
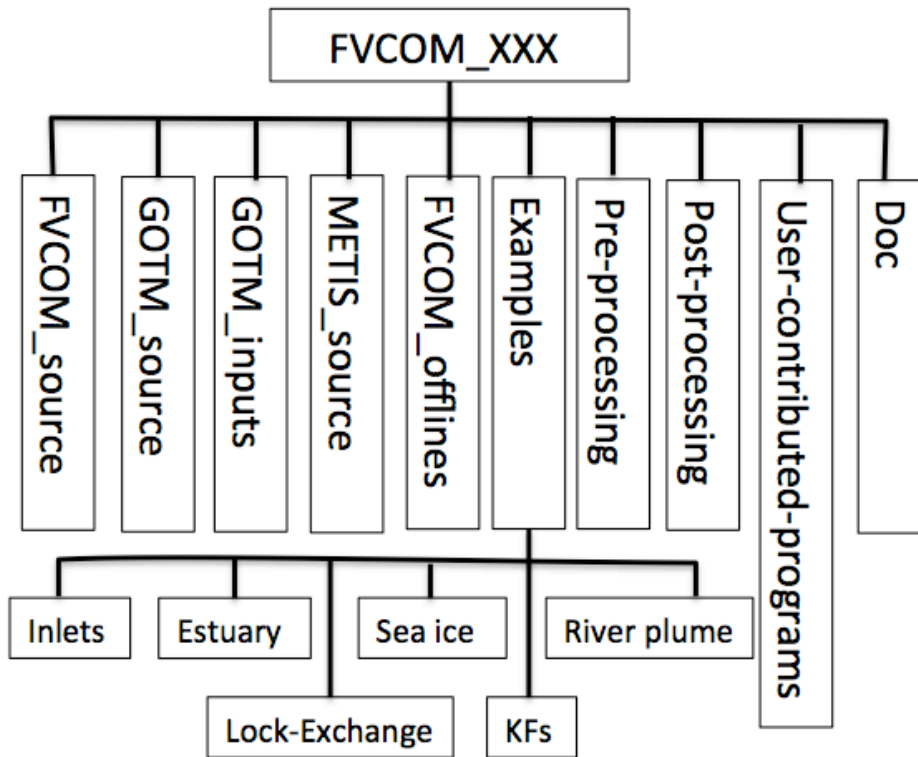
Figure 17.1: FVCOM directory structure

17.2a. Compiling METIS Libraries

(*For users intending to use the parallel code*) The METIS graph partitioning libraries are used to perform the domain decomposition portion of the FVCOM parallelization. These library routines are coded in the c language and must be compiled separately from the main coding. To compile the libraries, edit the makefile in the METIS source directory to point to your c compiler. It is advisable to use a c compiler from the same vendor as your intended Fortran compiler. This will help avoid compatibility problems when linking the libraries and the FVCOM code. To compile the libraries, use "make". A successful build will produce the file "libmetis.a" which will be linked during the FVCOM build. Note that if you do not intend to use FVCOM in a multiprocessor environment, it is not necessary to build these libraries.

17.2b. FVCOM Setup and Run

FVCOM 3.1.6 or higher is the new FVCOM version that includes many new modules, such as current-wave interaction, ocean-ice interaction, current-wave-sediment interaction, nesting modules over multi-domains and with other structured-grid models, and multi choices of different data assimilation methods.

To make this new code run more efficiently on multi-processor machines, we have significantly modified the code structure. In general, it requires five steps to set up and run FVCOM 3.1 or higher. They are:

1. Compile the source code,
2. Generate the run namelist,
3. Prepare input files,
4. Run FVCOM,
5. Monitor FVCOM performance.

A brief description of each step is given below.

A) Compiling the code

FVCOM includes two options for compilation of FVCOM: 1) automatically or 2) manually. For automatic compilation, one needs to run a shell file called "*configure.sh*" in the configuration folder to recreate the file called "*make.inc*".

For series run, type

<div align="center">./configure.sh series</div>

For a parallel run, type

<div align="center">./configure.sh parallel</div>

After the file "*make.inc*" is recreated, one compiles the FVCOM source code. This approach can help users to specify automatically "TOPDIR, "LIBDIR" and "INCIDR" (see the definitions below). The automatic compilation option has been tested for Linux and works well.

For manual setup, compilation of FVCOM is done using the "*make.inc*" included with the FVCOM source code in the FVCOM_source directory. Users must edit the "*make.inc*" to select desired code modules (parallelization, water quality, etc.) as well as

certain code options (e.g.,double/single precision). Then, variables in "*make.inc*" must be set to point to the Fortran compiler, the location of the *c*-preprocessor (*cpp*) as well as locations for the MPI (Message Passing Interface) libraries (if intending to run FVCOM in parallel) and the NetCDF libraries (if intending to use NetCDF output). In most cases, this can be completed through the following steps and remember, the "*make.inc*" is your friend.

STEP I: Go into the source code folder and find the file named "*make.inc*". Open "*make.inc*" to specify "TOPDIR", "LIBDIR" and "INCIDR", where

TOPDIR: the directory where "*make.inc*" is;

LIBDIR: the directory where libraries are installed;

INCIDR: the directory where include files are installed.

STEP II: Specify module flag-use "#" to uncomment module flags (removal of #) for the modules that you will not use. The value meanings are also outlined in the "*make.inc*". To avoid the deletion of "*make.inc*", we name "*make.inc*" in the FVCOM source code "*make.inc.example*". When using FVCOM, please first copy "*make.inc.example*" to "*make.inc*" and then edit it. The code module options to select before compilation are listed in Table 17.1.

Table 17.1: Code Module Options to Select before Compilation

Option	Values	Description
PRECISION	Activated	Chooses double precision for all floating point variables.
	Deactivated	Chooses single precision for all floating point variables.
SPHERICAL	Activated	Equations are solved in spherical coordinates. Note that the grid file must be in latitude/longitude format.
	Deactivated	Equations are solved in Cartesian coordinates.
FLOODING/DRYING	Activated	Include et/dry treatment of domain.
	Deactivated	Wet/dry treatment is not included.
MULTI_PROCESSOR	Activated	Code will include MPI-based parallelization. Note that MPI libraries must be linked at compile time.

	Deactivated	All parallelization will be removed from the code. No MPI libraries are necessary for compilation.
WATER_QUALITY	Activated	Water Quality Model (WQM) will be included in the compilation. If WQM is turned on in the runtime control file, the necessary WQM setup files must be present. See Chapter 11.
	Deactivated	WQM is not included
PROJECTION	Activated	Proj4 is turned on. Proj4 is the cartographic projection software developed by the USGS.
	Deactivated	Proj4 is not included.
DATA_ASSIMILATION	Activated	Nudging or OI-based data assimilation methods for SST, salinity and currents, are included.
	Deactivated	No nudging or OI data assimilation included
UPWIND LEAST SQUARE SCHEME LIMITER	Activated	Limit control is on. DLIMIIED_1: First order approximate limit; DLIMITED_2: Second order approximate limit.
	Deactivated	Limit is off
SEMI-IMPLICIT SCHEME	Activated	Semi-implicit solver is selected.
	Deactivated	Semi-implicit solver is not selected.
SOLID BOUNDARY (One and only one of these flags must be chosen.)	GCN	No ghost cells included.
	GCY1	Ghost cells included: ghost cell is located symmetrically relative to the boundary cell edge.
	GCY2	Ghost cells included: ghost cell is located symmetrically relative to the middle point of the boundary cell edge.
TURBULENCE MODEL	Activated	Use GOTM instead of the original FVCOM Mellor-Yamada 2.5 implementation for parameterization of vertical eddy viscosity.

	Deactivated	Use the original FVCOM Mellor-Yamada 2.5 implementation in FVCOM for parameterization of vertical eddy viscosity.
EQUILIBRIUM TIDE	Activated	Equilibrium tides included.
	Deactivated	No equilibrium tides included.
ATMOSPHERIC TIDE	Activated	Atmospheric tides included.
	Deactivated	No atmospheric tides included.
RIVER DISCHARGES	Activated	River discharges are included. When it is selected, the river discharge must be specified in the namelist and also input files.
	Deactivated	No rivers included.
MPDATA	Activated	FCT-multidimensional positive definite advection transport algorithm included for the vertical advection term. This algorithm was added by S. Hu, former graduate student at UMASSD. Use this algorithm with caution since it has not been well tested.
	Deactivated	Using original algorithm in FVCOM
TWO_D_MODEL	Activated	Set the model only for 2-D barotropic case.
	Deactivated	Set the model for both 2-D and 3-D modes (it is default setup).
BALANCE_2D	Activated	Output the terms in the 2-D momentum equations for the momentum balance analysis.
	Deactivated	No selection to output the terms in the 2-D momentum equations.
OPEN BOUNDARY TYPE OF WATER ELEVATION and TRANSPORT	Activated	Both water elevation and volume transport are specified. This flag is used for the case with no selection of nesting.
	Deactivated	Only water elevation is specified.

TIDAL OUTPUT FOR THE CASE WITH INCLUSION OF THE MEAN FLOW AT THE OPEN BOUNDARY	Activated	When the FLAG_18 is on, this selection will output the tidal information at boundary nodes and cells. This will create the input file to run FVCOM with inclusion of the mean flow at open boundary. The run is set up for the only tidal case. After the input file is created, then FLAG_18 could be selected and FLAG_19 is off.
	Deactivated	No meanflow boundary condition is selected.
DYE RELEASE	Activated	Turn on the dye release module.
	Deactivated	No dye release option selected.
SEDIMENT MODEL	Activated	Sediment model is included.
	Deactivated	No sediment model included.
KALMAN FILTERS	Activated	Kalman Filter data assimilation is selected. DRRKF: Reduced Kalman Filter. DENKF: Ensemble Kalman Filter.
	Deactivated	No Kalman Filter data assimilation selected.
1-D BIOLOGICAL MODEL	Activated	The 1-D biological model is selected. The 1-D refers to the case in which all physical and biological variables are uniform in the horizontal.

Users could select either a pre-built biological model (such as a NPZ and NPZD) or construct their own biological model using the pre-defined pool of biological variables and parameterization functions contained in the Generalized Ecosystem Module (GEM). |
| | Deactivated | No 1-D biological model selected. |

ICE MODEL	Activated	Turns on the sea ice module. The code was first set up for the spherical coordinate system and now it can be also used in the Cartesian coordinate system. When this option is selected, Flag_27 must be on.
	Deactivated	No sea ice model selected.
NET HEAT FLUX CALCULATION	Activated	When flag_26 is selected, this flag must be on to re-calculate the net heat flux with consideration of the ice-ocean interaction.
	Deactivated	When flag_26 is off, this option is not included.
AIR PRSSSURE	Activated	The air pressure gradient forcing is included.
	Deactivated	No air pressure gradient forcing included.
ViSiT ONLINE	Activated	Turns on ViSiT display program for the online monitoring of model performance.
	Deactivated	ViSiT is not set up with the model run.
NON-HYDROSTATIC MODEL	Activated	Non-hydrostatic solver is included.
	Deactivated	Non-hydrostatic solver is not included.
PARTICLE TRACKING	Activated	Particle tracking is turned on during the run. Note: it may slow down the computational efficiency. The FVCOM development team has already set up an offline particle tracking program, which is in the offline modules.
	Deactivated	Particle tracking during the run is turned off.
WAVE-CURRENT INTERACTION	Activated	Wave-current interaction module is turned on.
	Deactivated	Wave-current interaction is turned off.
DIKE and GROYNE MODEL	Activated	Dike and groyne module is turned on. Users are required to specify dikes and groynes in the dike-groyne input file.
	Deactivated	No dike and groyne module selected.

PWP MIXED LAYER MODEL	Activated	PWP mixed layer model is turned on. This option is used when the SST assimilation is selected. This helps avoid the formation of the thin stratified layer at the sea surface due to the SST assimilation. When no nudging or OI assimilation is selected, this flag should be off.
	Deactivated	No PWP mixed layer model selected.
VERTUCAL ADVECTION LIMITER	Activated	Vertical advection limiter is turned on. This module was implemented for the sigma-coordinate. For the s-coordinate case, it should be used in caution. Some modification might be needed.
	Deactivated	No vertical advection limiter selected.
PETSc	Activated	PETSc library is set up. This is required when a semi-implicit solver or non-hydrostatic solver or semi-implicit wave equation solver is selected.
	Deactivated	PETSc is off.
DEVELOPMEN FLAGS	Activated	This flag is used by the FVCOM development team for code testing. Users should always keep this flag commented.
	Deactivated	Development flag selection is off.
SELECT COMPILER/PLATFORM	Different compilers and platforms are defined for users' selection based on the computational environment they are using.	

Note: If you have compiled the model with a given set of options and you wish to change options you must recompile the entire code using "*make clean*" to remove all object files and executables and "*make*" to remake "*make.inc*".

Set variables in the "*make.inc*" for the location of your compiler and c preprocessor (*cpp*) as well as your compiler optimization/debugging flags. Variable descriptions and typical values are shown in Table 17.2. Note that you must use a Fortran90 or higher capable compiler. A listing of possible compilers and related information is provided in

Table 17.3. The c-preprocessor (*cpp*) is included in almost all Linux/Unix distributions as part of *gcc*. For Windows users, if *cpp* is not installed on the system, it can be made available through *cygwin* (see note to Windows users at beginning of chapter). Use "which *cpp*" to determine the location of *cpp* and supply the full path to the *makefile* variable CPP. The *makefile* currently contains several example sections for a variety of operating systems/compiler combinations. Uncomment a section if it is similar to your environment or provide values in the last section labeled "OTHER".

Table 17.2 Variables in the "*make.inc*"

Variable	Description	Note	Typical Value
CPP	C Preprocessor	Required	/usr/bin/cpp
CPPFLAGS	CPP Flags	Required	$(MOD_FLAGS)
FC	Fortran Compiler	Required	pgf90/xlf90/ifort/f90
OPT	Compiler Flags		Compiler + Machine Dependent
CLIB	Misc. Libraries		

Table 17.3 Fortran 90 Compilers for Clusters, Workstations, and Laptops

Compiler	Platforms	Type	FVCOM Tested	Notes
Intel v8	Windows/Linux	Commercial + Free Unsupported	Yes (Linux)	Outperforms Portland Group on Intel Chips
Portland Group	Windows/Linux	Commercial	Yes (Linux/Windows)	
Absoft	Windows/Linux	Commercial	No	
NAG	Windows/Linux	Commercial	No	
Lahey/Fujitsu	Windows/Linux	Commercial	No	

STEP III: Remove all object files and executables with "*make clean*". Compile code with "*make*". Watch screen output during compilation to ensure that the desired preprocessing flags are being utilized. Successful compilation will produce the executable *fvcom*. Copy or link the executable to the run directory. We recommend

linking because code change and subsequent recompilation will automatically be reflected in the executable in the run directory.

Note: If you want to use the fvcom pre-processing packages inside the FVCOM source code, you can access the directory called "input". For this case, you should compile the code by typing "*make all*" instead of "*make*".

B) Generating the namelist for the model run

STEP I: Make a folder "run" parallel to the FVCOM source folder. Copy or link the executable file "*fvcom*" to the run folder;

STEP II: Type "./fvcom", you will find the information about what is required to set up FVCOM. On the screen, you will see:

You must specify a case name:

Need to put something here!

This is not a very helpful help message!

LONG INPUT OPTIONS

--HELP => PRINT THIS MESSAGE

--CASENAME=<YOUR_CASE> (REQUIRED)

--CREATE_NAMELIST => PRINT BLANK NAMELIST AND RETURN

--LOGFILE=<FILENAME> => TO OUTPUT TO A LOG FILE

 --CRASHRESTART => RUN FROM CURRENT TIME IN RESTART FILE

 SHORT INPUT OPTIONS

 --V => PRINT FVCOM VERSION INFO AND RETURN

 --H => PRINT THIS MESSAGE AND RETURN

 DEBUG LEVELS

 --dbg=0 => DBG LOG (DEFAULT)

 --dbg=1 => DBG IO FILENAMES

 --dbg=2 => DBG SCALARS

 --dbg=4 => DBG SUBROUTINE NAMES

 --dbg=5 => DBG SUBROUTINE IO

 --dbg=6 => DBG VECTORS

 --dbg=7 => DBG EVERYTHING

--dbg_par => WRITE LOG FOR EACH PROCESSOR

STEP III: Type "./fvcom -- CREATE_NAMELIST blank, it will generate the blank namelist file named "*blank_run.nml*". Specify each term in "*blank_run.nml*" and rename it as your case name "*casename_run.nml*".   An example for selections of terms and parameters in "*blank_run.nml*" is given in Chapter 15.

C) Preparing input files

FVCOM 3.1 or up requires a different format for the input files from FVCOM 2.7. It uses two types of format. The first is the files with ASCII format, which include files named   "*casename_grd.dat*",   "*casename_dep.dat*",   "*casename_cor.dat*", "*casename_mc.dat*", and "*casename_bfw.dat*". The format of these files is very similar to those used for FVCOM 2.7, but header information is now required at the top of each file. For example, "Node Number = xxxx" and/or "Cell Number =xxxx". The file named "*casename_spg.dat*" requires the fixed header information of "Sponge Node Number = xx".   The second is the files with NetCDF format. Except for the ".dat" files listed above, all other input files must use the NetCDF format. For users who used to run FVCOM 2.7, they can go to the folder "input" and use the FVCOM built-in pre-processing package to convert old format files to new format files. We also provide the Matlab pre-processing programs to help users to create the NetCDF format input files for FVCOM 3.1 or up. These programs are in the FVCOM directory called "pre-processing".

D) Running FVCOM

Running FVCOM (Serial): The FVCOM executable "*fvcom*" takes only one argument, the casename = "*your casename*" or string used to identify a particular model. The general form of the executable command is

./fvcom --casename= *your casename*

For example, in our Gulf of Maine model, we have chosen "*gom*" as our casename. To execute the model, we type

./fvcom –casename= gom

on the command line.

Running FVCOM (Parallel): Parallel execution of FVCOM is machine dependent. If you are using a computer with a queuing system such as PBS or Load Leveler, seek details from the system manager. For desktop SMP's and small laboratory clusters with commercial or open source implementations of the MPI libraries, parallel execution will most likely be performed using the mpirun command script which comes with these distributions. A typical mpiexec startup of FVCOM to run a case might look like:

mpiexec –n "CPU number" ./fvcom –casename= *your casename*

Here the list of machines to be used (in order of processor allocation) is placed in the file "*mymachinefile*". The flag *n* takes as an argument the number of processors to run. Finally the executable *fvcom* and argument *casename* are supplied. Generally the machinefile list is simply a list of hostnames. For dual processor machines, the hostnames can be doubled up and in some cases the hostname can be appended with "-2". Read the manual supplied with your MPI distribution for further information.

Please note: FVCOM has been explicitly parallelized using the MPI Message Passing Libraries (Chapter 15; Cowles, 2008). This is considered a low-level parallelization because it is directly programmed into the code. A high-level parallelization is also possible through the compiler. Many modern compilers come with a flag that will instruct the compiler to perform a loop-based decomposition of the code while it is compiling. Afterwards the code can be executed in parallel using compiler specific commands and environmental variables. This method of parallelization is completely transparent to the user and can be performed directly on the serial (single processor) FVCOM code. However, this method of parallelization does not in general result in good parallel efficiency and will not scale well to a large number of processors. If you have a multi-processor workstation and you intend to normally execute the code on a single processor and occasionally on both processors, this may be a reasonable option because it does not require installation of the MPI libraries and parallel environment or compilation of the METIS libraries. Simply read the documentation for your compiler for auto parallelization and compile the serial FVCOM code with the appropriate flags.

E) Monitoring FVCOM run performance

FVCOM model results are stored in a NetCDF format. You can use the software "ViSiT" to view or visualize the model performance by plotting slides or making animations.

You can download ViSiT from https://wci.llnl.gov/codes/visit/ and install it following the installation manual. VISIT can be installed in Windows, Linux and Mac. To launch VISIT, for example, in Linux, type "visit". ViSiT has included functions that can directly read and display the FVCOM NetCDF files.

Chapter 18: Model Setup

The model must be given a case identification string, heretofore "casename", which will form the prefix for most input and output files. The string must be less than 80 characters and should not contain blanks or symbols. Models at SMAST use three letter acronyms (e.g. GOM gom = Gulf of Maine Model). The runtime control parameter file is named *casename_run.nml* and should be placed in the directory where the FVCOM executable file "*fvcom*" is located. All other input files should be placed in the directory which is specified in "*casename_run.nml*" with the name called "*INPUT_DIR*". Variables and parameters in "*casename_run.nml*" are described in section 18.1. Section 18.2 outlines the necessary input files for running particular cases.

18.1 FVCOM Runtime Control Parameter File *casename_run.nml*

After "*blank_run.nml*" is created, one needs to specify the parameters to control the FVCOM run. As a standard, one should be first rename this file as your case name "*casename_run.nml*" before editing it. This file contains all parameters used to control the execution of FVCOM. The file is designed to have a flexible format so that alignment of data values with particular columns is not necessary. This was done to eliminate incorrect reading of values, which can commonly occur with tab formatted input. Examples of the namelist file are included in the demo test cases distributed together with the code (in "*Examples*"). A brief description of the control parameters used in the "*.nml*" is given below. The items required to be specified are based on the selection of modules in the "*make.inc*".

1. Case description and time zone selection

```
&NML_CASE
 CASE_TITLE  = 'AN FVCOM CASE DESCRIPTION' - note string must be in 'quotes'
 TIMEZONE       = Select Time Zone or for idealized case select 'none' (start time=0.0)
 DATE_FORMAT  = A three letter string specify date format: 'YMD' or 'DMY'
 START_DATE  = Date and Time are specified as a string (example '2007-11-05 00:00:00')
 END_DATE  = For an idealized case specify 'seconds=(flt)','days=(flt)', or 'cycles=(int)'
```

Example for an idealized case run with no selection of year/day/hour/minutes/second:

```
&NML_CASE
 CASE_TITLE    = 'IDEAL ESTUARY CASE'
 TIMEZONE      = 'none'
 DATE_FORMAT    = 'YMD'
```

```
START_DATE    = 'days=0.'
END_DATE      = 'days=31.'
```

Example for the same case with selection of year/day/hour/minutes/seconds:

```
&NML_CASE
 CASE_TITLE     = 'IDEAL ESTUARY CASE'
 TIMEZONE       = 'UTC',
 DATE_FORMAT    = 'YMD'
 START_DATE     = '2010-08-30 00:00:00'
 END_DATE       = '2010-09-02 00:00:00'
```

2. Startup setup

```
&NML_STARTUP
 STARTUP_TYPE  = 'hotstart', 'coldstart', 'forecast' or 'crashrestart'
 STARTUP_FILE = blank_restart.nc
 STARTUP_UV_TYPE = 'default' or 'set values'
 STARTUP_TURB_TYPE   = 'default' or 'set values'
 STARTUP_TS_TYPE = 'constant' 'linear' 'observed' or 'set values'
 STARTUP_T_VALS  = 2*-99.00000
 STARTUP_S_VALS  = 2*-99.00000
 STARTUP_U_VALS  = -99.00000
 STARTUP_V_VALS  = -99.00000
 STARTUP_DMAX  = -99.00000
```

STARTUP_TYPE: The startup type with four options.	
coldstart	The model starts from a zero velocity field and forcings are ramped from zero to their full values over IRAMP.
hotstart	The model starts with a restart file from the model output as a selected time. This is usually used for the restart case with a NetCDF output available.
forecast	Like hotstart but the time for the restart is set up for forecast operation. In general, the forecast operation runs with the output of a series of restart files over given time intervals. Once "forecast" is selected and a restart time is specified, the model is back to find the restart data in the restart files at the exact same time.
crashrestart	Like hotstart but it searches for the latest restart file saved before the run crashed, and re-starts the model run automatically at that time.

STARTUP_FILE: The file used for the startup.	
Casename_restart.nc	The name of the restart file you like to use.
none	If "coldstart" is selected, no startup file is needed.

STARTUP_UV_TYPE: Specify the type of the startup u and v values.	
default	If "coldstart" is selected, fill in "default".
set values	If "hotstart" or "forecast" or "crashrestart" is select, use "set values". In this case, the model reads u and v values from the restart file.

STARTUP_TURB_TYPE: Specify types of turbulence mixing values.	
default	If "coldstart" is selected, fill in "default"
set values	If "hotstart" or "forecast" or "crashrestart" selected, fill in "set values". In this case, the model will start using turbulence variables and parameters from the restart file.

STARTUP_TS_TYPE: Specify types of the startup temperature (T) and salinity (S) values.	
constant	T and S remain constant during the run. When this option is selected, a value is required to fill in in "STARTUP_T_VALS" and STARTUP_S_VALS".
linear	The initial T and S change linearly in the vertical. When this option is selected, two values (at surface and bottom) are required to fill in in "STARTUP_T_VALS" and "STARTUP_S_VALS".
observed	This option is used to read the input T and S values at standard levels in the initial input file. If "coldstart" is selected, an initial input file for T and S is required. It also can be included at "hotstart". In this case, the T and S files need to be merged into the restart file.
set values	If "hotstart" is selected, the model will read T and S fields from the restart file.

STARTUP_T_VALS: Specify the startup temperature values.

When STARTUP_TS_TYPE is "constant", a single value is required. Since FVCOM ensures volume conservation in each control volume, the water temperature should remain unchanged during the model run, even for the case in which the temperature equation is solved. In general, you can give a value in the range between the freezing point of sea water (~ -2) to 40 (degrees C).

When STARTUP_TS_TYPE is "linear", two values are required, the surface and bottom temperatures, respectively. For example: 20 6. They present 20° at the surface and 6° at the bottom.

When STARUP_TS_TYPE is "observed" or "set values", this option will not be used in the code. Fill in a value you like. For example: 2*-99.00000, which is listed in the *casename_run.nml*.

STARTUP_S_VALS: Specify the startup salinity value

When STARTUP_TS_TYPE is "constant", a single value is required. Since FVCOM ensures volume conservation in each control volume, the water salinity should remain unchanged during the model run, even in the case in which the salinity equation is solved. At usual, you can give a value like "30".

When STARTUP_TS_TYPE is "linear", two values are required, the surface and bottom salinities, respectively. For example: 28 35. They present 28 psu at the surface and 35 psu at the bottom.

When STARUP_TS_TYPE is "observed" or "set values", this option will not be used in the code. Fill in a value you like. For example: 2*-99.00000, which is listed in the *casename_run.nml*.

STARTUP_U_VALS: Specify the startup x-coordinate component value of the velocity.

This option is rarely used. This allows users to specify the vertically uniform velocity at the initial time. For most applications, this option can be ignored.

STARTUP_V_VALS: Specify the startup y-coordinate component value of the velocity

See the above explanation.

STARTUP_DMAX: Specify the maximum water depth used in the T and S

interpolation for the initial condition. If it is in the water, it should be a negative value.

3. The IO Setup

```
&NML_IO
  INPUT_DIR  = /Your/relative/path/to/input/files
  OUTPUT_DIR = /Your/relative/path/to/output/files: Must already exist!
  IREPORT = 0
  VISIT_ALL_VARS  = F
  WAIT_FOR_VISIT  = F
  USE_MPI_IO_MODE = F
```

INPUT_DIR	Path connected to a sub-directory containing all input files
OUTPUT_DIR	Path connected to a sub-directory where the model output will be placed
IREPORT	Time interval (time steps) for screen displays of the computed values
VISIT_ALL_VARS	Set up ViSiT to monitor the FVCOM performance. "T" is on and "F" is off. If "T" is selected, ViSiT is online during the model run. If "F" is selected, ViSiT is off during the model run. We recommend selecting "F" to run the model more efficiently and then view the model results by reading the output data into ViSiT.
WAIT_FOR_VISIT	Waiting for ViSiT to connect before beginning integration. "T" is waiting and "F" is not waiting. We recommend selecting "F" to run the model more efficiently.
USE_MPI_IO_MODE	Specify the IO mode for the model output. If "T" is selected, the model uses one processor for output. If "F" is selected, the output uses the multi-processors. We recommend selecting "F".

An example could look like:

```
&NML_IO
INPUT_DIR = './input'
OUTPUT_DIR = './output'
IREPORT  =  40,
VISIT_ALL_VARS  = F
```

WAIT_FOR_VISIT = F,
USE_MPI_IO_MODE = F

4. Integration Setup

FVCOM includes two solvers: 1) mode-split and 2) semi-implicit. The namelist files for these two solvers are different.

An example for using the mode-split solver:

&NML_INTEGRATION
 EXTSTEP_SECONDS = 6.00
 ISPLIT = 10
 IRAMP = 500
 MIN_DEPTH = 0.05
 STATIC_SSH_ADJ = 0.0

An example for using the semi-implicit solver:

&NML_INTEGRATION
 INTSTEP_SECONDS = 60.0
 IRAMP = 500
 MIN_DEPTH = 0.05
 STATIC_SSH_ADJ = 0.0

EXTSTEP_SECOND	Time step for the external mode in seconds.
ISPLIT	Ratio of internal time step to external time step.
IRAMP	Number of time steps used to ramp the model from the initial field.
MIN_DEPTH	Minimum water depth used for the wet/dry treatment (unit: meters). A recommended depth is 0.05 m (5 cm). When no wet/dry treatment is selected, this parameter will not be used in the model run.
STATIC_SSH_ADJ	The static (or reference) depth adjustment. In many cases, the bathymetry was measured relative to the low water level. The water depth used in FVCOM is relative to the mean water level. For a small domain case, one can specify a constant value to adjust the water depth to the mean water level in the model coordinate system. For a regional domain, the mean sea level varies in space, so that one needs to adjust the

	bathymetric data to the reference depth using the mean sea level. It can be done by running the tidal simulation and adjusting the static water depth by adding the tide-predicted mean water level.
INTSTEP_SECONDS	Time step used for the semi-implicit solver. In general, we suggest that users use a time step based on the internal time step selected in the mode-split solver, even though a larger time step could still keep the model run stable. This approach could reduce the numerical damping that is hard to avoid with implicit solvers.

5. Restart Setup

&NML_RESTART
RST_ON = F
RST_FIRST_OUT= Date to start RESTART OUTPUT: Format the same as START_DATE
RST_OUT_INTERVAL = A length of time: 'seconds= ','days= ', or 'cycles= '
RST_OUTPUT_STACK = 0

RST_ON	Two choices: F (false) and T (true). If "T" is selected, the restart output file is on.
RST_FIRST_OUT	Date to start the restart output. The format is the same as "START_DATE" or could be the format described in RST_OUT_INTERNAL.
RST_OUT_INTERNAL	The time interval for the restart file output. It has three options: 1) "second= ", 2) "days= ", and 3) "cycles= ".
RST_OUTPUT_STACK	Specify the total number of the restart output file. If "0" is specified, the output will be a single file. If "1" is selected, at every restart output interval, a restart output file is created. If "2" is selected, one restart file will be created every two restart output intervals. If "3" is selected, the restart output files will be created every three restart output intervals, etc.

An example:

&NML_RESTART
 RST_ON = T,
 RST_FIRST_OUT   = '2009-06-01 00:00:00'
 RST_OUT_INTERVAL   = 'days = 31.'
 RST_OUTPUT_STACK   = 0

6. NetCDF Output Setup

&NML_NETCDF
 NC_ON = F
 NC_FIRST_OUT = Date to start NETCDF OUTPUT: Format the same as START_DATE
 NC_OUT_INTERVAL = A length of time: 'seconds= ','days= ', or 'cycles= '
 NC_OUTPUT_STACK = 0
 NC_SUBDOMAIN_FILES = 'filename for subdomain output'
 NC_GRID_METRICS = F
 NC_FILE_DATE   = F
 NC_VELOCITY   = F
 NC_SALT_TEMP   = F
 NC_TURBULENCE = F
 NC_AVERAGE_VEL = F
 NC_VERTICAL_VEL = F
 NC_NH_QP   = F
 NC_NH_RHS   = F
 NC_WIND_VEL   = F
 NC_WIND_STRESS = F
 NC_WAVE_PARA   = F
 NC_WAVE_STRESS = F
 NC_EVAP_PRECIP = F
 NC_SURFACE_HEAT = F
 NC_GROUNDWATER = F
 NC_ICE = F
 NC_BIO = F
 NC_WQM = F
 NC_VORTICITY   = F

NC_ON	NetCDF output controller with two options: F (false) and T (true). If "T" is selected, the output will be written as NetCDF format. If "F" is selected, no NetCDF output will be used.
RST_FIRST_OUT	Date to start the restart output. The format is the same as "START_DATE".
RST_OUT_INTERNAL	The time interval for the NetCDF output. It has three options: 1) "second= ", 2) "days= ", and 3) "cycles= ".
RST_OUTPUT_STACK	Specify the total number of the NetCDF output file. If "0" is specified, the output will be a single file. If "1" is selected, at every specified output interval, a NetCDF

	output file is created. If "2" is selected, one NetCDF file will be created every two specified output intervals. If "3" is selected, the NetCDF output files will be created every three specified output intervals, etc.
NC_SUBDOMAIN_FILES	This selection is used for the subdomain NetCDF output. If outputting the model results by selecting a subdomain, one needs to specify a filename in this selection. After this filename is selected, one needs to create the input file as the same filename to include the output modes and regions. The input files must contain 1) subdomain_mode –with options of "box" or "rds" (radius); 2) units -with options of "degree" or "meters", and 3) defined domain. See the example shown below.
NC_GRID_METRICS	Grid metrics output controller. "T" is on and "F" is off.
NC_FILE_DATE	The output time controller. "T" is on, which means that the time will be included in the NetCDF output file. "F" is off, which means that the time will not be included in the NetCDF output file.
NC_VELOCITY	3-D velocity output controller. "T" is on and "F" is off.
NC_SALT_TEMP	3-D salinity and temperature output controller. "T" is on and "F" is off.
NC_TURBULENCE	Turbulence variable output controller. "T" is on and "F" is off.
NC_AVERAGE_VEL	The 2-D velocity output controller. "T" is on and "F" is off.
NC_VERTICAL_VE	The vertical velocity output controller. "T" is on and "F" is off.
NC_NH_QP	The non-hydrostatic pressure output controller. "T" is on and "F" is off.
NC_NH_RHS	The right-hand side terms of the Poisson equation for the non-hydrostatic pressure output controller. "T" is on and

	"F" is off.
NC_WIND_VEL	The wind velocity output controller. "T" is on and "F" is off.
NC_WIND_STRESS	The surface wind stress output controller. "T" is on and "F" is off.
NC_WAVE_PARA	The surface wave parameters output controller. "T' is on and "F" is off. This selection needs to be chosen when the surface wave model is selected for the case with either only waves or current-wave interactions.
NC_WAVE_STRESS	The surface wave stress output controller. "T" is on and "F" is off.
NC_EVAP_PRECIP	Precipitation and evaporation output controller. "T" is on and "F" is off.
NC_SURFACE_HEAT	Surface heat flux output controller. "T" is on and "F" is off.
NC_GROUNDWATER	Groundwater output controller. "T" is on and "F" is off.
NC_ICE	Sea ice model result output controller. "T" is on and "F" is off.
NC_BIO	Biological model output controller. "T" is on and "F" is off.
NC_WQM	Water quality model output controller. "T" is on and "F" is off.
NC_VORTICITY	Vorticity output controller. "T" is on and "F" is off.

An example:

```
&NML_NETCDF
 NC_ON  = T
 NC_FIRST_OUT   = '2009-05-01 00:00:00'
 NC_OUT_INTERVAL = 'seconds=3600.'
 NC_OUTPUT_STACK = 0
 NC_SUBDOMAIN_FILES = 'AF447'
 NC_GRID_METRICS = T
 NC_FILE_DATE   = F
 NC_VELOCITY    = T
 NC_SALT_TEMP   = T
 NC_TURBULENCE  = T
```

```
 NC_AVERAGE_VEL  = F
 NC_VERTICAL_VEL = T
 NC_WIND_VEL    = T
 NC_WIND_STRESS  = F
 NC_EVAP_PRECIP  = F
 NC_SURFACE_HEAT = F
 NC_GROUNDWATER  = F
 NC_ICE  = F
 NC_BIO  = F
 NC_WQM  = F
 NC_VORTICITY   = F
```

7. NetCDF Output Setup for Time Averaged Results

```
&NML_NETCDF_AV
NCAV_ON = F
NCAV_FIRST_OUT=Date to start NETCDF interval averaged output: Format the same as START_DATE,
NCAV_OUT_INTERVAL = A length of time: 'seconds= ','days= ', or 'cycles= '
NCAV_OUTPUT_STACK = 0
NCAV_SUBDOMAIN_FILES   = FVCOM
NCAV_GRID_METRICS = F
NCAV_FILE_DATE  = F
NCAV_VELOCITY   = F
NCAV_SALT_TEMP  = F
NCAV_TURBULENCE = F
NCAV_AVERAGE_VEL = F
NCAV_VERTICAL_VEL = F
NCAV_NH_QP  = F
NCAV_NH_RHS = F
NCAV_ICE  = F
NCAV_WIND_VEL   = F
NCAV_WIND_STRESS      = F
NCAV_WAVE_PARA  = F
NCAV_WAVE_STRESS = F
NCAV_EVAP_PRECIP  = F
NCAV_SURFACE_HEAT = F
NCAV_GROUNDWATER = F
NCAV_BIO  = F
NCAV_WQM = F
NCAV_VORTICITY  = F
```

The definition for each item listed above is the same as those listed in "NetCDF Output Setup", except items here are for the output of time-averaged model variables. Here NCAV_OUT_INTERVAL refers to the averaging time length.

An example:

```
&NML_NETCDF_AV
 NCAV_ON = T
 NCAV_FIRST_OUT  = '2009-05-01 00:00:00'
 NCAV_OUT_INTERVAL     = 'days=1.'
 NCAV_OUTPUT_STACK     = 0
 NCAV_SUBDOMAIN_FILES   = 'FVCOM'
 NCAV_GRID_METRICS     = T
```

```
NCAV_FILE_DATE  = F
NCAV_VELOCITY   = T
NCAV_SALT_TEMP  = T
NCAV_TURBULENCE = T
NCAV_AVERAGE_VEL      = T
NCAV_VERTICAL_VEL     = T
NCAV_ICE      = T
NCAV_WIND_VEL  = F
NCAV_WIND_STRESS     = F
NCAV_EVAP_PRECIP     = F
NCAV_SURFACE_HEAT    = F
NCAV_GROUNDWATER     = F
NCAV_BIO     = F
NCAV_WQM     = F
NCAV_VORTICITY  = F
```

8. Surface Forcing Setup

```
&NML_SURFACE_FORCING
 WIND_ON = F
 WIND_TYPE= Options::speed,stress
 WIND_FILE = example_wnd.nc
 WIND_KIND= Options:constant,static,time dependant,periodic,variable
 WIND_X =  0.0000000E+00
 WIND_Y =  0.0000000E+00
 HEATING_ON  = F
 HEATING_TYPE = 'body' or 'flux'
 HEATING_KIND = Options:constant,static,time dependant,periodic,variable
 HEATING_FILE  = example_wnd.nc
 HEATING_LONGWAVE_LENGTHSCALE = 1.400000
 HEATING_LONGWAVE_PERCTAGE = 0.7800000
 HEATING_SHORTWAVE_LENGTHSCALE = 6.300000
 HEATING_RADIATION = 0.0000000E+00
 HEATING_NETFLUX = 0.0000000E+00
 PRECIPITATION_ON = F
 PRECIPITATION_KIND = Options:constant,static,time dependant,periodic,variable
 PRECIPITATION_FILE  = example_emp.nc
 PRECIPITATION_PRC =  0.0000000E+00
 PRECIPITATION_EVP =  0.0000000E+00
 AIRPRESSURE_ON  = F
 AIRPRESSURE_KIND = Options:constant,static,time dependant,periodic,variable
 AIRPRESSURE_FILE  = example_aip.nc
 AIRPRESSURE_VALUE  =  0.0000000E+00
 WAVE_ON = F
 WAVE_FILE  = example_wav.nc
 WAVE_KIND = Options:constant,static,time dependant,periodic,variable
 WAVE_HEIGHT =  0.0000000E+00
 WAVE_LENGTH =  0.0000000E+00
 WAVE_DIRECTION =  0.0000000E+00
 WAVE_PERIOD =  0.0000000E+00
 WAVE_PER_BOT  =  0.0000000E+00
 WAVE_UB_BOT =  0.0000000E+00
```

WIND_ON	Wind forcing controller. "T" is on and "F" is off.

WIND_TYPE	Wind forcing types. Two options: "speed" or "stress".
WIND_FILE	Filename for wind forcing input file in a NetCDF format
WIND_KIND	Wind forcing kinds. Five options: 1) "constant", 2) "static", 3) "time dependent", 4) "periodic" and 5) "variable". "constant": the wind is uniform over the entire computational domain but not in time; "static": the wind varies in space but not in time; "time dependent": the wind varies in time but not in space; "periodic": the wind varies periodically in time; "variable": the wind varies in both space and time.
WIND_X	The *x*-component of 10-m wind velocity (units: m/s) or surface wind stress (unit: N/m^2). This option is only used for the wind kind = constant.
WIND_Y	The *y*-component of 10-m wind velocity (unit: m/s) or surface wind stress (unit: N/m^2). This option is only used for the wind kind = constant.
HEATING_ON	Surface heat flux controller. "T" is on and "F" is off.
HEATING_TYPE	Heating flux types. There are two choices: "body" or "flux". For the selection of "body", the shortwave radiation in the water column is specified directly as a body forcing in the advection term of the temperature equation. For the selection of "flux", the shortwave radiation in the water column is added into the model in the vertical diffusion term as layer fluxes.
HEATING_KIND	Heating kinds. Five options: 1) "constant", 2) "static", 3) "time dependent", 4) "periodic" and 5) "variable". "constant": the surface heat flux is uniform over the entire computational domain but not in time; "static": the surface heat flux varies in space but not in

	time;
	"time dependent": the surface heat flux varies in time but not in space;
	"periodic": the surface heat flux varies periodically;
	"variable": the surface heat flux varies in space and time.
HEATING_FILE	Filename of the surface heat flux input file in a NetCDF format. Heat flux data are input as a component of meteorological forcing as a single NetCDF file.
HEATING_LONGWAVE _LENGTHSCALE	The attenuation length for the longer wavelength component of shortwave irradiance (ZETA1)
HEATING_LONGWAVE _PERCTAGE	The fraction of the total shortwave flux associated with the longer wavelength irradiance (RHEAT).
HEATING_SHORTWAV E_LENGTHSCALE	The attenuation depth for shorter wavelength component of shortwave irradiance (ZETA2)
HEATING_RADIATION	The shortwave irradiance (watts/m^2). This option is only used when HEATING_KIND is "constant". In this case, ZETA1, ZETA2 and RHEAT need to be specified.
HEATING_NETFLUX	The net surface heat flux (unit: watts/m^2). This option is only used when HEATING_KIND is "constant".
PRECIPITATION_ON	Precipitation (P) minus evaporation (E) controller. "T" is on and "F" is off.
PRECIPITATION_KIND	P-E kinds: "constant", "static", "time dependent", "periodic", and "variable".
PRECIPITATION_FILE	P-E input filename in a NetCDF format. This option is used when PRECIPITATION_KIND is "variable".
PRECIPITATION_PRC	Precipitation value if PRECIPITATION_KIND is "constant".
PRECIPITATION_EVP	Evaporation value if PRECIPITATION_KIND is "constant".
AIRPRESSURE_ON	The air pressure gradient controller. "T" is on and "F" is

	off.
AIRPRESSURE_KIND	The air pressure kinds: "constant", "static", "time dependent", "periodic", and "variable".
AIRPRESSURE_FILE	The filename of the air pressure input file in a NetCDF format when AIRPRESSURE_KIND is "variable".
AIRPRESSURE_VALUE	The air pressure value (units: Pa; N/m^2; kg m^{-1} s^{-2}). In most cases, it can be specified as a standard constant atmospheric air pressure value of 101325.00 Pa. This option is only used when AIRPRESSURE_KIND is "constant".

Example:

```
&NML_SURFACE_FORCING
 WIND_ON = T
 WIND_TYPE = 'speed'
 WIND_FILE = 'wrf_wind-2009.nc'
 WIND_KIND  = 'variable'
 WIND_X  =  0.0000000E+00
 WIND_Y  =  0.0000000E+00
 HEATING_ON   = T
 HEATING_TYPE = 'flux'
 HEATING_KIND  = 'variable'
 HEATING_FILE   = 'heat-2009.nc'
 HEATING_LONGWAVE_PERCTAGE      =  0.78
 HEATING_LONGWAVE_LENGTHSCALE   =  1.4
 HEATING_SHORTWAVE_LENGTHSCALE  =  6.3
 HEATING_RADIATION   =  0.0
 HEATING_NETFLUX = 0.0000000E+00
 PRECIPITATION_ON  = T
 PRECIPITATION_KIND = 'variable'
 PRECIPITATION_FILE  = 'prate-2009.nc'
 PRECIPITATION_PRC  =  0.0000000E+00
 PRECIPITATION_EVP  =  0.0000000E+00
 AIRPRESSURE_ON  = T
 AIRPRESSURE_KIND      = 'variable'
 AIRPRESSURE_FILE      = 'press-2009.nc'
 AIRPRESSURE_VALUE      =  0.0000000E+00
```

9. One-way SWAVE_FVCOM Coupling

```
&NML_Oneway_SWAVE_FVCOM_Coupling
 WAVE_ON = F
 WAVE_FILE  = example_wav.nc
 WAVE_KIND = Options:constant,static,time dependant,periodic,variable
 WAVE_HEIGHT =  0.0000000E+00
```

```
 WAVE_LENGTH =  0.0000000E+00
 WAVE_DIRECTION =  0.0000000E+00
 WAVE_PERIOD =  0.0000000E+00
 WAVE_PER_BOT  =  0.0000000E+00
 WAVE_UB_BOT =  0.0000000E+00
```

In FVCOM 3.1.6 or up, we include an option to run FVCOM with given wave fields, and call this as "one-way" wave-current coupling. In this case, users can run the wave model separately and input the wave field into FVCOM. Such a coupling does allow us to consider the wave-current interaction process but no feedback to the wave simulation is considered. To run a one-way wave-current coupling experiment, one activates "FLAG_37 = "WAVE_OFFLINE", in the "*make.inc*".

WAVE_ON	Surface wave model controller. "T" is on and "F" is off.
WAVE_FILE	The surface wave variable and parameter input file in a NetCDF format.
WAVE_KIND	Wind forcing used to drive the surface wave model. Two options: "constant" and "variable".
WAVE_HEIGHT	The value of significant wave height when WAVE_KIND is "constant".
WAVE_LENGTH	The value of wavelength when WAVE_KIND is "constant".
WAVE_DIRECTION	The value of wave direction when WAVE_KIND is "constant".
WAVE_PERIOD	The value of wave period when WAVE_KIND is "constant".
WAVE_PRE_BOT	The value of bottom wave period when WAVE_KIND is "constant".
WAVE_UB_BOT	The value of bottom orbital velocity when WAVE_KIND is "constant".

An example:

```
&NML_Oneway_SWAVE_FVCOM_Coupling
WAVE_ON = T
WAVE_FILE = 'yangze_wav.nc'
WAVE_KIND = 'variable'
WAVE_HEIGHT    = 0.00000
WAVE_LENGTH    = 0.00000
WAVE_DIRECTION  = 0.00000
```

WAVE_PERIOD   = 0.00000
WAVE_PER_BOT   = 0.00000
WAVE_UB_BOT   = 0.00000

10. Option to Calculate Heat Flux

&NML_HEATING_CALCULATED
 HEATING_CALCULATE_ON:
 HEATING_CALCULATE_TYPE:
 HEATING_CALCULATE_FILE:
 HEATING_CALCULATE_KIND:
 ZUU:
 ZTT:
 ZQQ:
 AIR_TEMPERATURE:
 RELATIVE_HUMIDITY:
 SURFACE_PRESSURE:
 LONGWAVE_RADIATION:
 SHORTWAVE_RADIATION:
 HEATING_LONGWAVE_PERCTAGE_IN_HEATFLUX:
 HEATING_LONGWAVE_LENGTHSCALE_IN_HEATFLUX:
 HEATING_SHORTWAVE_LENGTHSCALE_IN_HEATFLUX:

HEATING_CALCULATE_ON	Heat flux calculation option controller. "T" is on and "F" is off.
HEATING_CALCULATE_TYPE	Heat flux types. Two options: "body" or "flux". See the surface forcing for their definitions.
HEATING_CALCULATE_FILE	The recalculated surface heat flux input file in a NetCDF format.
HEATING_CALCULATE_KIND	The heat flux kinds. Five options: "constant", "static", "time dependent", "periodic", and "variable".
ZUU	The height of wind measurement above the sea surface (unit: meters).
ZTT	The height of air temperature measurement above the sea surface (unit: meters).
ZQQ	The height of relative humidity measurement above the sea surface (unit: meters).
AIR_TEMPERATURE	The value of air temperature. This option is only used when HEATING_CALCULATE_KIND is

	"constant".
RELATIVE_HUMIDITY	The value of relative humidity. This option is only used when HEATING_CALCULATE_KIND is "constant".
SURFACE_PRESSURE	The value of surface air pressure. This option is only used when HEATING_CALCULATE_KIND is "constant".
LONGWAVE_RADIATION	The value of long-wave radiation (watts/m^2) when HEATING_CALCULATE_KIND is "constant".
SHORTWAVE_RADIATION	The value of short-wave radiation (watts/m^2) when HEATING_CALCULATE_KIND is "constant".
HEATING_LONGWAVE_PERCTAGE_IN_HEATFLUX	The fraction of the total shortwave flux associated with the longer wavelength irradiance (RHEAT).
HEATING_LONGWAVE_LENGTHSCALE_IN_HEATFLUX	The attenuation length for the longer wavelength component of the shortwave irradiance (ZETA1)
HEATING_SHORTWAVE_LENGTHSCALE_IN_HEATFLUX	The attenuation depth for shorter wavelength component of the shortwave irradiance (ZETA2)

11. Physical Parameter Setup

```
&NML_PHYSICS
HORIZONTAL_MIXING_TYPE  = 'closure' or 'constant'
HORIZONTAL_MIXING_FILE  = example_hvc.nc
HORIZONTAL_MIXING_KIND  = Options:constant,static
HORIZONTAL_MIXING_COEFFICIENT   = -1.000000
HORIZONTAL_PRANDTL_NUMBER     = -1.000000
VERTICAL_MIXING_TYPE   = 'closure' or 'constant'
VERTICAL_MIXING_COEFFICIENT    = -1.000000
VERTICAL_PRANDTL_NUMBER = -1.000000
BOTTOM_ROUGHNESS_TYPE = 'orig', or 'gotm'; Select your bottom roughness equation (brough.F)
BOTTOM_ROUGHNESS_KIND = Options:constant,static
BOTTOM_ROUGHNESS_FILE = example_brf.nc
BOTTOM_ROUGHNESS_LENGTHSCALE  = -1.000000
BOTTOM_ROUGHNESS_MINIMUM  = -1.000000
CONVECTIVE_OVERTURNING  = F
```

SCALAR_POSITIVITY_CONTROL = F
BAROTROPIC = F
BAROCLINIC_PRESSURE_GRADIENT = 'sigma levels' or 'z coordinates'; select method of calculation
SEA_WATER_DENSITY_FUNCTION= 'dens1', 'dens2', or 'dens3; Select your equation of state
(eqs_of_state.F)
RECALCULATE_RHO_MEAN = F
INTERVAL_RHO_MEAN = A length of time or number of cycles in standard format
TEMPERATURE_ACTIVE = F
SALINITY_ACTIVE = F
SURFACE_WAVE_MIXING = F
WETTING_DRYING_ON = F
ADCOR_ON = T
EQUATOR_BETA_PLANE = F
NOFLUX_BOT_CONDITION = T

HORIZONTAL_MIXING_TYPE	Horizontal diffusion types. Two options: "closure" or "constant". "closure": The horizontal diffusion is calculated using the Smagorinsky's parameterization method. "constant": The horizontal diffusion coefficient needs to be specified in "HORCON".
HORIZONTAL_MIXING_FILE	The input horizontal diffusion filename that allows reading in the horizontal diffusion coefficient from the input file. If no input file is specified, write in "none".
HORIZONTAL_MIXING_KIND	Horizontal diffusion kinds. Two options: 1) "constant" and 2) "static". When HORIZONTAL_MIXING_TYPE is "closure", the horizontal diffusion is calculated using the Smagorinsky parameterization method, in which a coefficient needs to be specified. This coefficient is not the mixing

	coefficient, and it can be a constant or non-uniform in space ("static").
HORIZONTAL_MIXING_COEFFICIENT	The value of the horizontal diffusion coefficient (unit: m^2/s) when HORIZONTAL_MIXING_KIND is "constant". The mixing coefficient presents a different meaning for "closure" and "constant" selections of HORIZONTAL_MIXING_TYPE. For "closure", this coefficient represents a constant used in the Smagorinsky parameterization. For "constant, this coefficient represent a real horizontal mixing coefficient.
HORIZONTAL_PRANDTL_NUMBER	The horizontal Prandtl number (HPRNU). This is the ratio of horizontal thermal diffusion to horizontal eddy viscosity. HPRNU is included as 1/HPRNU in the horizontal eddy viscosity equation of the momentum equation. If HPRNU is specified as a value of 0.1, then it means that the horizontal eddy viscosity will be amplified by 10 times.
VERTICAL_MIXING_TYPE	Vertical mixing types. Two options: 1) "closure" and 2) "constant".
VERTICAL_MIXING_COEFFICIENT	The value of the vertical eddy viscosity (units: m^2/s) when VERTICAL_MIXING_TYPE is "constant".

VERTICAL_PRANDTL_NUMBER	Vertical Prandtl number (VPRNU), which is defined as a ratio of vertical thermal diffusion to vertical eddy viscosity. VPRNU is included in the thermal diffusion term in the temperature or salinity equation. If VPRNU is specified to be 0.1, it means that the thermal diffusion is reduced by 10 times.
BOTTOM_ROUGHNESS_TYPE	Bottom roughness types. Two options: 1) "orig" and 2) "gotm". One can select a bottom roughness equation in "brough.F".
BOTTOM_ROUGHNESS_KIND	Bottom roughness kinds. Two options: 1) "constant" and 2) "static".
BOTTOM_ROUGHNESS_FILE	The bottom roughness input filename. Users can read in a spatially varying bottom roughness field as an input fie. If no input file is specified, write in "none".
BOTTOM_ROUGHNESS_LENGTHSCALE	Bottom roughness length scale when BOTTOM_ROUGHNESS_KIND is "constant".
BOTTOM_ROUGHNESS_MINIMUM	Minimum bottom roughness value. In many case, it is set up for a default value of 0.0025 m.
CONVECTIVE_OVERTURNING	Convection adjustment controller. "T" is on and "F" is off.
SCALAR_POSITIVITY_CONTROL	Positive scalar value controller. "T" is on and "F" is off. When "T" is selected, the model will turn on a

	controller to make sure all scalar variables remain positive.
BAROTROPIC	Barotropic experiment controller. "T" is to set up a barotropic experiment, and "F" means "no".
BAROCLINIC_PRESSURE_GRADIENT	Selection of the methods to calculate the baroclinic pressure gradient. Two options: 1) "sigma levels" and 2) "z-coordinate". If a generalized terrain following coordinate is selected, "sigma-levels" means that the baroclinic pressure gradient is calculated in that coordinate.
SEA_WATER_DENSITY_FUNCTION	Selection of the seawater density function. Three choices: "dens1", "dens2", and "dens3".
RECALCULATE_RHO_MEAN	Controller for recalculation of the mean density field. "T" is on and "F" is off.
INTERVAL_RHO_MEAN	The time interval for recalculation of the mean density field. It has three options: 1) "second= ", 2) "days= " and 3) "cycles= ".
TEMPERATURE_ACTIVE	Controller for water temperature calculation. "T" is to solve the temperature equation, and "F" is off.
SALINITY_ACTIVE	Controller for water salinity calculation. "T" is to solve the salinity equation, and "F" is off.
SURFACE_WAVE_MIXING	Surface wave mixing controller. "T" is to consider the parameterized surface

	wave mixing in the level 2.5 MY turbulence model and "F" is off.
WETTING_DRYING_ON	Wet/dry treatment controller. "T" is to set up the wet/dry treatment and "F" is off.
ADCOR_ON	Coriolis term correction controller. "T" is to calculate the Coriolis term in the momentum equation semi-implicitly, and "F" is to calculate this term explicitly.
EQUATOR_BETA_PLANE	Beta plane approximation controller. "T" is to calculate the pressure gradient in a range of 5°N and 5°S using the beta plane approximation, and "F" is off.

An example:

```
&NML_PHYSICS
 HORIZONTAL_MIXING_TYPE        = 'closure'
 HORIZONTAL_MIXING_KIND        = 'constant'
 HORIZONTAL_MIXING_COEFFICIENT  = 0.40
 HORIZONTAL_PRANDTL_NUMBER      = 0.10
 VERTICAL_MIXING_TYPE        = 'closure'
 VERTICAL_MIXING_COEFFICIENT    = 1.0E-6
 VERTICAL_PRANDTL_NUMBER      = 1.0
 BOTTOM_ROUGHNESS_MINIMUM      = 0.0025
 BOTTOM_ROUGHNESS_LENGTHSCALE   = 0.001
 BOTTOM_ROUGHNESS_KIND        = 'constant'
 BOTTOM_ROUGHNESS_TYPE        = 'orig'
 CONVECTIVE_OVERTURNING        = F
 SCALAR_POSITIVITY_CONTROL      = T
 BAROTROPIC            = F
 BAROCLINIC_PRESSURE_GRADIENT    = 'sigma levels'
 SEA_WATER_DENSITY_FUNCTION     = 'dens3'
 RECALCULATE_RHO_MEAN        = F
 INTERVAL_RHO_MEAN        = 'days=20.'
 TEMPERATURE_ACTIVE        = T
 SALINITY_ACTIVE          = T
 SURFACE_WAVE_MIXING        = F
 WETTING_DRYING_ON        = F
 ADCOR_ON              = T
 EQUATOR_BETA_PLANE        = F
```

12. River Input Setup

```
&NML_RIVER_TYPE
RIVER_NUMBER   =        -1
RIVER_KIND      = Options:periodic or variable
RIVER_TS_SETTING      = 'calculated' or 'specified'
RIVER_INFO_FILE = 'default' or 'filename'
RIVER_INFLOW_LOCATION   = 'node' or 'edge'
/
&NML_RIVER
 RIVER_NAME  = River Name in netcdf data file; use mulitple namelists for multiple rivers!,
 RIVER_FILE = example_split_riv.nc
 RIVER_GRID_LOCATION  = -1
RIVER_VERTICAL_DISTRIBUTION   = 100*-99.00000
```

RIVER_NUMBER	Total number of river discharge points (number of rivers).
RIVER_KIND	The river input kinds. Two options: 1) "periodic" and 2) "variable". The selection of "periodic" allows us to run the model by repeating the same river discharge. The selection of "variable" also can include the constant case.
RIVER_TS_SETTING	The methods used to calculate water salinity or temperature at the discharge point. The are two choices: 1) "calculated" - the salinity or temperature at discharge nodes is calculated using the mass conservative salinity or temperature equation, and 2) "specified" - the salinity or temperature at discharge nodes is specified by users.
RIVER_INFO_FILE	The filename of the river input namelist. Two options: 1) "default" and 2) "filename".
RIVER_INFLOW_LOCATION	The location of the river inflow. Two options: 1) "node"-discharge water is added from node points as point sources and 2)

	"edge"-discharge water is added from boundary lines of triangles.
RIVER_NAME	The river name in the NetCDF river input file. Use "*River_mamelist.nml*" for multiple rivers.
RIVER_FILE	The river discharge input filename. The input file requires a NetCDF format. This can be defined in "*River_Namelist.nml*" for a multiple river case.
RIVER_GRID_LOCATION	The node or cell numbers where the river discharge is added. This can be defined in "*River_Namelist_nml*" for a multiple river case.
RIVER_VERTICAL_DISTRIBUTION	The vertical proportion of the river discharge.

An example:

```
&NML_RIVER_TYPE
 RIVER_NUMBER = 709
 RIVER_TS_SETTING = 'calculated'
 RIVER_INFLOW_LOCATION = 'edge'
 RIVER_INFO_FILE = 'RIVERS_NAMELIST.nml'
 RIVER_KIND = 'variable'
```

An example of the file "*River_Namelist.nml*":

```
&NML_RIVER
 RIVER_NAME           = 'river1'
 RIVER_FILE           = 'River_data.nc'
 RIVER_GRID_LOCATION      = 1569
 RIVER_VERTICAL_DISTRIBUTION = 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 /

 &NML_RIVER
 RIVER_NAME           = 'river2'
 RIVER_FILE           = 'River_data.nc'
 RIVER_GRID_LOCATION      = 1567
 RIVER_VERTICAL_DISTRIBUTION = 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 /

 &NML_RIVER
 RIVER_NAME           = 'river3'
```

```
RIVER_FILE             = 'River_data.nc'
RIVER_GRID_LOCATION        = 1565
RIVER_VERTICAL_DISTRIBUTION = 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 /
```

13. Open Boundary Setup

```
&NML_OPEN_BOUNDARY_CONTROL
 OBC_ON = F
 OBC_NODE_LIST_FILE = example_split_obc.dat
 OBC_ELEVATION_FORCING_ON= F
 OBC_ELEVATION_FILE = example_split_obc.nc
 OBC_TS_TYPE    = -1
 OBC_TEMP_NUDGING = F
 OBC_TEMP_FILE  = example_split_obc.nc
 OBC_TEMP_NUDGING_TIMESCALE =  0.0000000E+00
 OBC_SALT_NUDGING = F
 OBC_SALT_FILE   = example_split_obc.nc
 OBC_SALT_NUDGING_TIMESCALE=  0.0000000E+00
 OBC_WQM_NUDGING = F
 OBC_WQM_FILE   = example_split_obc.nc
 OBC_WQM_NUDGING_TIMESCALE=  0.0000000E+00
 OBC_BIO_NUDGING = F
 OBC_BIO_FILE   = example_split_obc.nc
 OBC_BIO_NUDGING_TIMESCALE=  0.0000000E+00


 OBC_MEANFLOW    = F
 OBC_MEANFLOW_FILE = example_split_obc.nc
 OBC_TIDEOUT_INITIAL= 0
 OBC_TIDEOUT_INTERVAL    = 0
 OBC_LONGSHORE_FLOW_ON   = F
 OBC_LONGSHORE_FLOW_FILE = example_split_lsf.dat
```

OBC_ON = F,	Open boundary condition controller. "T" is on and "F" is off.
OBC_NODE_LIST_FILE	The input file of the open boundary node number. This selection is also applied for equilibrium tidal case. In the makefile, when tidal equilibrium tide and obc file is selected, the filename of OBC_NODE_LIST_FILE is included.
OBC_ELEVATION_FORCING_ON	Controller for the sea level open boundary conditions. "T" is on and "F" is off. When "T" is selected, an input NetCDF filename for

	tidal boundary condition is required to given in "OBC_ELEVATION_FILE". The input data can be either amplitudes and phases of tidal constituents at nodes (non-Julian) or the time series of the tidal elevation at nodes using Julian day. The tidal forcing kinds (Julian or non-Julian) are defined in the NetCDF input file.
OBC_ELEVATION_FILE	The input file of the boundary tidal elevation (in a NetCDF format). When an obc file is created and the equilibrium tide is selected, A OBC_ELEVATION_FILE is created, which includes tidal periods, equilibrium tidal amplitudes and love numbers.
OBC_TS_TYPE	Temperature (T) and salinity (S) open boundary condition types. A detail is given in "TYPE_TSOBC" of the " *mod_obcs*". The T and S at open boundaries can be determined in several ways. For a pure open boundary, T and S are first divided into a sum of vertically averaged and perturbation values. Then two steps calculate T and S at open boundaries. The first step is to calculate the vertically averaged heat and salt flux in the volume conservative boundary control volumes. The second step is to use the radiation boundary condition methods to calculate the perturbation values at open boundaries. There are four options to calculate the perturbation of T and S at open boundaries. They are: TYPE_TSOBC = 1: No gradient condition;

	TYPE_TSOBC = 2: Implicit gravity-wave radiation condition. TYPE_TSOBC = 3: Blumberg&Kantha implicit radiation condition. TYPE_TSOBC = 4: Orlanski radiation condition.
OBC_TEMP_NUDGING	Controller for temperature nudging boundary condition. "T" is on and "F" is off. No matter how good the numerical accuracy is, the radiation condition can produce the loss of heat and salt because all values at open boundaries are determined from the interior. To retain the heat and salt in the computational domain, we also create the boundary condition that allows us to specify T and S for the inflow. The temperature nudging only functions during the inflow period.
OBC_TEMP_FILE	The input file that contains the temperature values at open boundary nodes. This file is required if OBC_TEMP_NUDGING is selected.
OBC_TEMP_NUDGING_TIMESCALE	The time scale coefficient for temperature nudging. It is defined as 1/(time steps larger than one hour in time). Example: if the time step is 30 seconds and a recovery time is 3 hours (a nudging time scale), the time scale could be $1/(3\times3600/30)=2.8E\text{-}3$.
OBC_SALT_NUDGING	Controller for salinity nudging boundary condition. "T" is on and "F" is off.

OBC_SALT_FILE	The input file that contains the salinity values at open boundary nodes. This file is required if OBC_SALT_NUDGING is selected.
OBC_SALT_NUDGING_TIMESCALE	The time scale coefficient for salinity nudging.
OBC_WQM_NUDGING	Controller for water quality nudging boundary condition. "T" is on and "F" is off.
OBC_WQM_FILE	The input file that contains the water quality values at open boundary nodes. This file is required if OBC_WQM_NUDGING is selected.
OBC_WQM_NUDGING_TIMESCALE	The time scale coefficient for water quality nudging.
OBC_BIO_NUDGING	Controller for biological variables nudging boundary condition. "T" is on and "F" is off.
OBC_BIO_FILE	The input file that contains the biological variables values at open boundary nodes. This file is required if OBC_BIO_NUDGING is selected.
OBC_BIO_NUDGING_TIMESCALE	The time scale coefficient for biological variables nudging.
OBC_MEANFLOW	Controller for adding the mean flow at open boundaries. "T" is on and "F" is off. When "T" is selected, in the *make.inc*, FLAG = MEAN_FLOW needs to be uncommented.
OBC_MEANFLOW_FILE	The input file of tidal boundary forcing in a NetCDF format.
OBC_TIDEOUT_INITIAL	The initial time at which the model starts outputting tidal simulation results. When the "*meanflow*" is included, one needs to run the model for only tidal forcing first to create the

	boundary tidal forcing file. This file is read in through "OBC_MEANFLOW_FILE". When the model runs with inclusion of the "*meanflow*" tidal boundary, this option is not needed anymore. For example: TIDE_INITIAL = '2005-06-10 00:00:00'
OBC_TIDEOUT_INTERVAL	The time interval for tidal model output. For example: TIDE_INTERVAL = 'seconds=720.0'
OBC_LONGSHORE_FLOW_ON	Controller for including the along-shore flow at the open boundary. "T" is on and "F" is off. This is designed for the Gulf of Maine FVCOM runs. Users should not use it unless having the same inflow condition as the Gulf of Maine.
OBC_LONGSHORE_FLOW_FILE	The filename for the input file that contain the along-shore inflow information.

An example for tidal boundary with Julian Day:

```
&NML_OPEN_BOUNDARY_CONTROL
 OBC_ON = T
 OBC_NODE_LIST_FILE = 'tst_obc.dat'
 OBC_ELEVATION_FORCING_ON = T
 OBC_ELEVATION_FILE = 'julian_obc.nc'
```

An example for tidal boundary with Julian Day:

```
&NML_OPEN_BOUNDARY_CONTROL
 OBC_ON = T
 OBC_NODE_LIST_FILE = 'tst_obc.dat'
 OBC_ELEVATION_FORCING_ON = T
 OBC_ELEVATION_FILE = 'm2_only_1m.nc'
```

An example for global FVCOM with selection of equilibrium tides:

```
NML_OPEN_BOUNDARY_CONTROL
OBC_ON = F
OBC_NODE_LIST_FILE = 'glbn_obc.dat'
```

OBC_ELEVATION_FORCING_ON = T
OBC_ELEVATION_FILE = 'spectral_obc.nc'
OBC_TS_TYPE = 3
OBC_TEMP_NUDGING = F
OBC_TEMP_FILE = 'none'
OBC_TEMP_NUDGING_TIMESCALE = 0.0000000E+00
OBC_SALT_NUDGING   = F
OBC_SALT_FILE = 'none'
OBC_SALT_NUDGING_TIMESCALE = 0.0000000E+00
OBC_MEANFLOW = F
OBC_MEANFLOW_FILE = 'spec_ideal.nc'

An example used for ECS-FVCOM with inclusion of Kuroshio transport on southern and northern open boundaries:

&NML_OPEN_BOUNDARY_CONTROL
 OBC_ON   = T
 OBC_NODE_LIST_FILE   = 'ecsbigger_obc.dat'
 OBC_ELEVATION_FORCING_ON   = T
 OBC_ELEVATION_FILE   = 'julian_obc.nc'
 OBC_TS_TYPE   = 3
 OBC_TEMP_NUDGING   = F
 OBC_TEMP_FILE   = 'none'
 OBC_TEMP_NUDGING_TIMESCALE = 0.0000000E+00,
 OBC_SALT_NUDGING   = F
 OBC_SALT_FILE   = 'none'
 OBC_SALT_NUDGING_TIMESCALE = 0.0000000E+00,
 OBC_MEANFLOW   = T
 OBC_MEANFLOW_FILE   = 'meanflow_tide.nc'

14. Grid Coordinates Setup

&NML_GRID_COORDINATES
 GRID_FILE   = example_split_grd.dat
 GRID_FILE_UNITS = Can be 'degrees' or 'meters'; certain make options required
 PROJECTION_REFERENCE = none: A recognized reference coordinate for projection for PROJ4
 SIGMA_LEVELS_FILE = example_split_sigma.dat
 DEPTH_FILE   = example_split_dep.dat
 CORIOLIS_FILE = example_split_cor.dat
 SPONGE_FILE   = example_split_spg.dat

GRID_FILE	The grid input filename.
GRID_FILE_UNITS	Unit used for measuring the distance in the model. Two options: 1) "degree" and 2) "meters".
PROJECTION_REFERENCE	The recognized reference coordinate used for projection if PROJ4 is on.
SIGMA_LEVELS_FILE	The vertical coordinate input filename.
DEPTH_FILE	The water depth input file in an ASCII format.

CORIOLIS_FILE	The latitude input file (that is used to calculate Coriolis parameter at each node) in an ASCII format.
SPONGE_FILE	The sponge layer input filename. This file contains parameters used to define a sponge layer for damping purpose at open boundaries.

An example used for global-FVCOM:

```
&NML_GRID_COORDINATES
 GRID_FILE      = 'glbn_grd.dat'
 GRID_FILE_UNITS = 'degrees'
 SIGMA_LEVELS_FILE    = 'glbn_sig_5m.dat'
 DEPTH_FILE     = 'glbn_dep.dat'
 CORIOLIS_FILE   = 'glbn_cor.dat'
 SPONGE_FILE    = 'glbn_spg.dat'   %Note: Global-FVCOM has no open boundary. This file
                                          will not used.
```

An example used for global-FVCOM:

```
&NML_GRID_COORDINATES
 GRID_FILE      = 'ecsbigger_grd.dat'
 GRID_FILE_UNITS = 'degrees'
 PROJECTION_REFERENCE  = 'proj=tmerc +datum=NAD83 +lon_0=123d10 lat_0=30d50
k=.9999666666666667 x_0=0 y_0=0'
 SIGMA_LEVELS_FILE    = 'ecsbigger_sigma.dat'
 DEPTH_FILE     = 'ecsbigger_dep.dat'
 CORIOLIS_FILE   = 'ecsbigger_cor.dat'
 SPONGE_FILE    = 'ecsbigger_spg.dat'
```

15. Groundwater Setup

```
&NML_GROUNDWATER
 GROUNDWATER_ON  = F
 GROUNDWATER_TEMP_ON    = F
 GROUNDWATER_SALT_ON    = F
 GROUNDWATER_KIND  = Options:constant,static,time dependant,periodic,variable
 GROUNDWATER_FILE = example_grndwtr.nc
 GROUNDWATER_FLOW = 0.0000000E+00
 GROUNDWATER_TEMP = 0.0000000E+00
 GROUNDWATER_SALT = 0.0000000E+00
```

GROUNDWATER_ON	Controller for groundwater input. "T" is on and "F" is off.
GROUNDWATER_TEMP_ON	Controller for groundwater temperature input. "T" is on and "F' is off.

GROUNDWATER_SALT_ON	Controller for groundwater salinity input. "T" is on and "F' is off.
GROUNDWATER_KIND	Kinds of groundwater input. Four options: "constant", "static", "time dependent", and "variable".
GROUNDWATER_FILE	The filename for groundwater input.
GROUNDWATER_FLOW	The groundwater flux value if GROUNDWATER_KIND is "constant".
GROUNDWATER_TEMP	The groundwater temperature value if GROUNDWATER_KIND is "constant".
GROUNDWATER_SALT	The groundwater salinity value if GROUNDWATER_KIND is "constant".

An example without inclusion of groundwater input:

```
&NML_GROUNDWATER
 GROUNDWATER_ON           = F
 GROUNDWATER_FLOW  = 0.0
 GROUNDWATER_FILE        = 'none'
 /
```
An example with groundwater input:

```
&NML_GROUNDWATER
 GROUNDWATER_ON  = T
 GROUNDWATER_TEMP_ON = T
 GROUNDWATER_SALT_ON = T
 GROUNDWATER_KIND = 'variable'
 GROUNDWATER_FILE = 'groundwater.nc'
 GROUNDWATER_FLOW = 0.0
 GROUNDWATER_TEMP = 0.0
 GROUNDWATER_SALT = 0.0
```

16. Lagrangian Particle Tracking Setup

```
&NML_LAG
LAG_PARTICLES_ON  = F
LAG_START_FILE  = init_lag.nc
LAG_OUT_FILE    = lag_out.nc
LAG_FIRST_OUT   = A date or time
LAG_RESTART_FILE  = lag_restart.nc
LAG_OUT_INTERVAL  = A length of time: 'seconds= ','days= ', or 'cycles= '
LAG_SCAL_CHOICE = none
```

/

LAG_PARTICLES_ON	Controller for the online Lagrangian particle tracking. "T" is on and "F" is off.
LAG_START_FILE	The filename for the input file that contains number and initial locations of particles.
LAG_OUT_FILE	The filename for the output of particle tracking locations with time in a NetCDF format.
LAG_FIRST_OUT	The initial time at which the locations of particles are written into the output file.
LAG_RESTART_FILE	The NetCDF format restart filename
LAG_OUT_INTERVAL	The time interval for the output, which can be "second=", "days=", and "cycles=".
LAG_SCAL_CHOICE	Selection of scalar variables to include with output. These scalar variables include salinity, temperature, density, vertical eddy viscosity and vertical thermal diffusion coefficient. Users must set variable name in "get_ud_scalnames".

An example:

```
&NML_LAG
 LAG_PARTICLES_ON      = T
 LAG_START_FILE  = 'lag_init.nc'
 LAG_OUT_FILE    = 'lag_out.nc'
 LAG_FIRST_OUT    = "cycle=0"
 LAG_RESTART_FILE = 'none'
 LAG_OUT_INTERVAL =  'seconds=60.0'
 LAG_SCAL_CHOICE  = 'SALINITY'
```

17. Data Assimilation, Biological Model, Sediment Model and Ice Model Setup

```
&NML_ADDITIONAL_MODELS
DATA_ASSIMILATION  = F
DATA_ASSIMILATION_FILE  = ./example_split_run.nml
BIOLOGICAL_MODEL  = F
STARTUP_BIO_TYPE = 'observed' use this option only at current code.
SEDIMENT_MODEL  = F
SEDIMENT_MODEL_FILE = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF FIRST
SEDIMENT_PARAMETER_TYPE= DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF
FIRST
```

SEDIMENT_PARAMETER_FILE = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF FIRST
 BEDFLAG_TYPE = DO NOT ADD UNTIL FVCOM IS RUNNING BY ITS SELF FIRST
 BEDFLAG_FILE   = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF FIRST
 ICING_MODEL = F
 ICING_FORCING_FILE = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF FIRST
 ICING_FORCING_KIND = Options:constant,static,time dependant,periodic,variable
 ICING_AIR_TEMP = 0.0000000E+00
 ICING_WSPD = 0.0000000E+00
 ICE_MODEL = F
 ICE_FORCING_FILE = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF FIRST
 ICE_FORCING_KIND = Options:constant,static,time dependant,periodic,variable
 ICE_SEA_LEVEL_PRESSURE = 0.0000000E+00
 ICE_AIR_TEMP   = 0.0000000E+00
 ICE_SPEC_HUMIDITY   = 0.0000000E+00
 ICE_SHORTWAVE = 0.0000000E+00
 ICE_CLOUD_COVER = 0.0000000E+00

DATA_ASSIMILATION	Controller for nudging or OI data assimilation. "T" is on and "F" is off.
DATA_ASSIMILATION_FILE	The namelist file to define the selected data assimilation. In general, the filename looks like "casename_run.nml".
BIOLOGICAL_MODEL	Controller for turning on an online biological model. "T" is on and "F" is off.
STARUP_BIO_TYPE	The input file that defines parameters controlling the initial condition of biological variables. Now we only have one option of "observed".
SEDIMENT_MODEL	Controller for turning on the sediment model. "T" is on and "F" is off.
SEDIMENT_MODEL_FILE	The filename of the sediment model input file
SEDIMENT_PARAMETER_TYPE	Define the parameters controlling the distribution of the critical shear stress for erosion and deposition as well as the erosion rate. Two options: 1) "constant" and 2) "non-uniform".

SEDIMENT_PARAMETER_FILE	The filename of the input file that contains the non-uniform distributions of shear stress and erosion rate. The file requires a NetCDF format.
BEDFLAG_TYPE	Flag used to activate or deactivate erosion and deposition processes at a given node in the sediment model. Two options: 1) BEDFLAG= 1: deactivate erosion and deposition and 2) BEDFLAG=2: activate erosion and deposition.
BEDFLAG_FILE	The filename of the bedflag input files to specify spatial distribution of bed processes using flag. This file requires a NetCDF format.
ICING_MODEL	Controller for turning on/off the icing calculation. "T" is on and "F" is off.
ICING_FORCING_FILE	The filename of the sea surface boundary condition input file in a NetCDF format. This file contains specific humidity, air temperature and cloud cover. The wind speed will be provided in the wind forcing input for FVCOM.
ICING_FORCING_KIND	The icing forcing kinds. Five options: "constant", "static", "time dependent", "periodic" and "variable". If "variable" is selected, the "ICING_FORCING_FILE" is required.
ICING_AIR_TEMP	The sea surface air temperature value if the "constant" forcing kind is selected.
ICING_WSPD	The 10-m wind speed (unit: m/s) value if the "constant" forcing kind is selected.
ICE_MODEL	Controller for turning on/off the ice model

	UG-CICE. "T" is on and "F" is off.
ICE_FORCING_FILE	The filename of the ice model input file that contains the net heat flux, shortwave irradiance, etc. This file requires a NetCDF format. The air pressure field is required in the surface forcing selection.
ICE_FORCING_KIND	The ice forcing kinds. Five options: "constant", "static", "time dependent", "periodic" and "variable".
ICE_SEA_LEVEL_PRESSURE	The sea surface air pressure value if the ICE_FORCING_KIND is "constant".
ICE_AIR_TEMP	The air temperature value if ICE_FORCING_KIND is "constant".
ICE_SPEC_HUMIDITY	The specific humidity if ICE_FORCING_KIND is "constant".
ICE_SHORTWAVE	The solar shortwave irradiance value if ICE_FORCING_KIND is "constant".
ICE_CLOUD_COVER	The cloud cover value if ICE_FORCING_KIND is "constant".

An example for data assimilation and ice model setup;

```
&NML_ADDITIONAL_MODELS
 DATA_ASSIMILATION      = T
 DATA_ASSIMILATION_FILE = glbn_run.nml
 BIOLOGICAL_MODEL       = F
 STARUP_BIO_TYPE   = 'observed'
 SEDIMENT_MODEL  = F,
 SEDIMENT_MODEL_FILE    = 'none'
 ICING_MODEL    = F
 ICING_FORCING_FILE    = 'none'
 ICING_FORCING_KIND     = 'variable'
 ICING_AIR_TEMP  =  0.0000000E+00
 ICING_WSPD     = 0.0000000E+00
 ICE_MODEL      = T
 ICE_FORCING_FILE       = 'ice_forcing-2009.nc'
 ICE_FORCING_KIND       = 'variable'
```

```
ICE_SEA_LEVEL_PRESSURE  =  0.0000000E+00
ICE_AIR_TEMP    =  0.0000000E+00
ICE_SPEC_HUMIDITY      =  0.0000000E+00
ICE_SHORTWAVE  =  0.0000000E+00
ICE_CLOUD_COVER =  0.0000000E+00
 /
```

An example for deactivating and activating bedflag:

Option 1 - No BEDFLAG (bed processes active everywhere)
BEDFLAG_TYPE = 'constant'
BEDFLAG_FILE = 'none'

Option 2 - Specify spatial distribution of bed processes using flag
BEDFLAG_TYPE = 'variable'
BEDFLAG_FILE = skg4.3_bedflag.nc'

18. Station Output Setup

```
&NML_PROBES
 PROBES_ON      = F
 PROBES_NUMBER  = 0
 PROBES_FILE  = Probe namelist file name
```

PROBES_ON	Controller for turning on/off station outputs to separate files. "T" is on and "F" is off. In this case, the model will create an output file for each selected station.
PROBES_NUMBER	Number of stations
PROBES_FILE	The filename of the probe namelist file that contains variables the list of output variables.

The times series output module contains subroutines which setup the variable for time series output and dump them to a file at times specified by the user in the setup file. Time series output is activated by setting PROBE_ON = T in the *casename_run.nml*. Probe setup files are specified in the probe namelist file.

19. Station Time Series Output Setup

```
&NML_STATION_TIMESERIES
 OUT_STATION_TIMESERIES_ON     = F
 STATION_FILE   = 'none'
 LOCATION_TYPE  = 'node' or 'cell'
 OUT_ELEVATION  = F
 OUT_VELOCITY_3D = F
```

```
OUT_VELOCITY_2D = F
OUT_WIND_VELOCITY     = F
OUT_SALT_TEMP   = F
OUT_INTERVAL    = A length of time: 'seconds= ','days= ', or 'cycles= '
```

OUT_STATION_TIMESERIES_ON	Controller for turning on/off station time series output to a single file. "T" is on and "F" is off.
STATION_FILE	The filename for the input station file that contains the station node or cell numbers. This is an ascii format file.
LOCATION_TYPE	Location types. Two options: 1) "node" and 2) "cell".
OUT_ELEVATION	Controller for the sea elevation time series output at selected stations. "T" is on and "F" is off.
OUT_VELOCITY_3D	Controller for the 3-D velocity time series output at selected stations. "T" is on and "F" is off.
OUT_VELOCITY_2D	Controller for the 2-D velocity time series output at selected stations. "T" is on and "F" is off.
OUT_WIND_VELOCITY	Controller for the wind velocity time series output at selected stations. "T" is on and "F" is off.
OUT_SALT_TEMP	Controller for the salinity and temperature time series outputs at selected stations. "T" is on and "F" is off.
OUT_INTERVAL	The output time interval. It can be "second=", "days=", or "cycle=".

20. Variable Bound Check Setup

```
&NML_BOUNDSCHK
 BOUNDSCHK_ON   = F
```

```
CHK_INTERVAL  = 0
VELOC_MAG_MAX  =  0.0000000E+00
ZETA_MAG_MAX  =  0.0000000E+00
HS_MAX  =  1.0000000E+09
TEMP_MAX  =  0.0000000E+00
TEMP_MIN  =  0.0000000E+00
SALT_MAX  =  0.0000000E+00
SALT_MIN  =  0.0000000E+00
/
```

BOUNDSCHK_ON	Controller for turning on/off the variable bound check. "T" is on and "F" is off.
CHK_INTERVAL	The checking time interval, which is defined as time step number. For example, if one want to check these parameter every 5 time steps, then CHK_INTERVAL = 5.
VELOC_MAG_MAX	The maximum absolute value of velocity.
ZETA_MAG_MAX	The maximum value of elevation.
HS_MAX	The maximum value of significant wave height.
TEMP_MAX	The maximum value of water temperature.
TEMP_MIN	The minimum value of water temperature.
SALT_MAX	The maximum value of water salinity.
SALT_MIN	The minimum value of water salinity.

21. Boundary Output Setup for FVCOM

```
&NML_NCNEST
 NCNEST_ON = F
 NCNEST_BLOCKSIZE = -1
 NCNEST_NODE_FILES = none
```

NCNEST_ON	Controller for turning on/off the nesting boundary output in the large model domain. "T" is on and "F" is off.
NCNEST_BLOCKSIZE	The block size of the nesting boundary output data at an output time. This specification allows users to save the model data in many blocks and output

	more efficiently. The selection of this number depends on the integration time and grid size. For example, if "200" is specified, this means that the model will save all output variables of every 200 time integration into a block and written into the output file one time.
NCNEST_NODE_FILES	The filename of the nesting boundary input file that contains the information of the nesting boundary nodes. Format: ASCII.

An example for global-FVCOM nesting boundary output for NECOFS:

```
&NML_NCNEST
NCNEST_ON = T
NCNEST_BLOCKSIZE =  200
NCNEST_NODE_FILES =  'glbn_sub_nest_node.dat'
```

22. Nesting Boundary Input Setup for FVCOM

```
&NML_NESTING
 NESTING_ON  = F
 NESTING_TYPE  = '1' or '2' or '3'
 NESTING_BLOCKSIZE  = -1
 NESTING_FILE_NAME = example_split_nesting.nc
```

NESTING_ON	Controller for turning on/off the nesting subdomain model run. "T' is on and "F" is off.
NESTING_TYPE	Nesting types. Three options: "1": Direct nesting -- The input is the direct NCNEST output of large domain FVCOM model; "2": Indirect nesting -- Same as "1" except the input file is the subtidal values from the NCNEST output of large domain. For this case, the subdomain model will retain its own tidal forcing at the nesting boundary. When this option is selected, Forman's tidal analysis program will be

	turned on to remove the subdomain model-predicted subtidal elevations at nodes and velocity at cells in the nesting boundary zone. "3": Relaxing nesting -- This is used for nesting FVCOM with a structured-grid model. This module was configured for nesting FVCOM with global-HYCOM. It can be modified to be used for other structured-grid models.
NESTING_BLOCKSIZE	The block size of the nesting boundary data input at a time step.
NESTING_FILE_NAME	The filename of the nesting boundary input file that contain all required nesting variables. This file requires a NetCDF format.

21. Nesting Boundary Output Setup for SWAVE

```
&NML_NCNEST_WAVE
 NCNEST_ON_WAVE  = F
 NCNEST_TYPE_WAVE  = 'wave parameters' or 'spectral density'
 NCNEST_BLOCKSIZE_WAVE   = -1
 NCNEST_NODE_FILES_WAVE  = none
```

NCNEST_ON_WAVE	Controller for turning on/off the wave nesting output in the large domain. "T" is on and "F" is off.
NCNEST_TYPE_WAVE	The wave nesting types. Two options: 1) "wave parameters" and 2) "spectral density". FVCOM includes two methods to nesting multi-domain SWAVE operations. For "wave parameters", the large domain SWAVE will output the significant wave height and peak period, and specify these values at nesting boundary nodes. For "spectral density", the large domain SWAVE will output all spectral values and specify them as nesting boundary conditions.

NCNEST_BLOCKSIZE_WAVE	The block size of the wave model output data at one output time. See explanation in "NESTING_BLOCKSIZE".
NCNEST_NODE_FILES_WAVE	The filename of the nesting boundary input file that contains the information of the nesting boundary nodes. Format: ASCII.

22. Nesting Boundary Input Setup for SWAVE

```
&NML_NESTING_WAVE
 NESTING_ON_WAVE = F
 NESTING_TYPE_WAVE  = 'wave parameters' or 'spectral density'
 NESTING_BLOCKSIZE_WAVE  = -1
 NESTING_FILE_NAME_WAVE  = example_nesting_wave.nc
```

NESTING_ON_WAVE	Controller for turning on/off the wave nesting subdomain model run. "T' is on and "F" is off.
NESTING_TYPE_WAVE	The wave nesting types. Two options: 1) "wave parameters" and 2) "spectral density". FVCOM includes two methods to nesting multi-domain SWAVE operations. For "wave parameters", the large domain SWAVE will output the significant wave height and peak period, and specify these values at the nesting boundary nodes. For "spectral density", the large domain SWAVE will output all spectral values and specify them as nesting boundary conditions.
NESTING_BLOCKSIZE_WAVE	The block size of the wave model output data at one output time. See explanation in "NESTING_BLOCKSIZE".
NESTING_FILE_NAME_WAVE	The filename of the nesting boundary input file that contains all variables used for specifying

	the nesting boundary condition. This file requires a NetCDF format.

23. Semi-implicit Solver Setup

&NML_SEMI
IFCETA = 0.5500000
BEDF = 1.000000
KSTAGE_UV = 1
KSTAGE_TE = 1
KSTAGE_TS = 1
MSTG = slow

IFCETA	Implicit factor for the surface pressure gradient term, ifceta = 0 for a fully explicit time stepping method; ifceta=1 for a fully implicit time stepping method. A recommended value is 0.55.
BEDF	Factor to enforce a zero normal flux condition at a solid wall boundary in the free surface matrix inversion. Factor can be specified from a value of 1 (no allowing the flux across a solid wall boundary) to a value of 0 (allowing the flux across a solid wall boundary).
KSTAGE_UV	Factor to set up the Nth step multistage explicit advection scheme to calculate the advection terms when a semi-implicit solver is selected. "1" represents the 1st stage scheme which is identical to an explicit advection scheme, "2" is the 2nd stage scheme, and "N" is the Nth stage scheme. Increasing the stage number allows a user to choose a longer time step with a sacrifice in computational efficiency at each integration loop. The default setup is "1".
KSTAGE_TE	Factor to set up the Nth step multistage explicit advection scheme to solve the turbulence advection terms in the turbulence closure models when a semi-implicit scheme is selected. See "KSTAGE_UV" for explanation on how a factor could be selected. A default setup is "1".
KSTAGE_TS	Factor to set up the Nth step multistage explicit advection scheme to solve the advection terms in temperature or salinity equations when a semi-implicit scheme is selected. See "KSTAGE_UV" for

	explanation on how a factor could be selected. The default setup is "1".
MSTG	A flag to control the computational efficiency of the multistage advection scheme. Two options: 1) "slow" and 2) "fast". When "fast" is chosen, the solver will try to use a novel way to speed up the multistage explicit advection scheme in the momentum equation. This option still needs more testing, so we don't recommend a new user to use it. The default selection is "slow".

24. Dye Release Setup

```
&NML_DYE_RELEASE
 DYE_ON = F
 DYE_RELEASE_START = Date or time to start dye release: Format the same as START_DATE
 DYE_RELEASE_STOP = Date or time to stop dye release: Format the same as START_DATE
 KSPE_DYE = 0
 MSPE_DYE = 0
 K_SPECIFY = 100*0
 M_SPECIFY = 200*0
 DYE_SOURCE_TERM = 1.000000
```

DYE_ON	Controller for turning on/off the dye release. "T" is on and "F" is off.
DYE_RELEASE_START	Date or time to start dye release in the same format as "START_DATE".
DYE_RELEASE_STOP	Date or time to stop dye tracking in the same format as "START_DATE".
KSPE_DYE	The total number of vertical layers in which the dye will be released.
MSPE_DYE	The total number of nodes where the dye will be released.
K_SPECIFY	The index number of layers in which the dye will be released. For example, suppose we configure FVCOM with a total of 31 vertical levels. Then KB=31 and the total layer number is KB-1=30. If we plan to release the

	dye in just the three layers above the bottom, then KSPE_DYE = 3 and K_SPECIFY = 28, 29, 30.
M_SPECIFY	The node numbers where the dye will be released. For example, we plan to release dye at three nodes of a triangle, then MSPE_DYE = 3, and M_SPECIFY = num1, num2, num3. Here num1, num2 and num3 are the node numbers of that triangle.
DYE_SOURCE_TERM	Dye concentration

```
&NML_WATERQUALITY
WATER_QUALITY_MODEL = F
WATER_QUALITY_MODEL_FILE  = DO NOT ADD UNTILL FVCOM IS RUNNING BY ITS SELF
FIRST
BENWQM_KEY = F
STARTUP_WQM_TYPE = 'constant' 'linear' 'observed' or 'set values'
STARTUP_WQM_VALS = 16*-99.00000
```

WATER_QUALITY_MODEL	Controller for turning on/off the online water quality model. "T" is on and "F" is off. **Note**: the FVCOM team has implemented three water quality models. See Chapter 11 for information. This online water quality model was used for Georgia estuaries when FVCOM 2.4 was released. No further tests were done for FVCOM 3.1.6 or up.
WATER_QUALITY_MODEL_FILE	The filename of the water quality model input file for initial conditions.
BENWQM_KEY	Controller for turning on/off the benthic flux. "T" is on and "F" is off.
STARTUP_WQM_TYPE	Types of the water quality initial condition. Four options: 1) "constant", 2) "linear", 3) "observed" and "set values".
STARTUP_WQM_VALS	The initial values of water quality variables if STARTUP_WQM_TYPE is "constant".

	The values include:
	1) Dissolved oxygen (DO) (mg O2/l);
	2) Carbonaceous biochemical oxygen demand
	(CBOD) (mg C/l);
	3) Phytoplankton (PHYT) (mg C/l);
	4) Ammonia Nitrogen (NH3), (mg N/l),
	5) Nitrate Nitrogen (NO3) (mg N/l),
	6) Organic Nitrogen (ON) (mg N/l);
	7) Orthophosphorus (or inorganic phosphorus,
	OPO4) (mg P/l)
	8) Organic phosphorus (OP), (mg P/l).
	These values can be input together separated
	by a space.

25. Non-hydrostatic solver setup

&NML_NH
PROJ_SWITCH =-1

PROJ_SWITCH	Selection of the non-hydrographic pressure solver. Two options: 1) "Projection" and 2) "Pressure correction". This needs to be an integer number. The default value is "-1" for the choice of the pressure correction method. To choose the projection method, the integer should be a number larger than the last integration time step.

26. PWP Mixing Layer Model Setup for SST Assimilation

&NML_PWP
UPPER_DEPTH_LIMIT=20.00000
LOWER_DEPTH_LIMIT=200.0000
VERTICAL_RESOLUTION=1.000000
BULK_RICHARDSON=0.6500000
GRADIENT_RICHARDSON=0.2500000

In FVCOM, the SST assimilation is performed through the surface mixed layer to avoid the formation of an unreal thin thermal layer. We implemented the PWP surface mixed layer to determine the mixed depth which is then used as the vertical range for the SST assimilation.

UPPER_DEPTH_LIMIT	The minimum water depth where PWP model is applied. If the local water depth is less than this depth, the PWP model will be turned off.
LOWER_DEPTH_LIMIT	The maximum mixed layer depth at which the PWP model calculation is turned off. Mixing in the PWP model is determined by three criterions. Set up this maximum mixed layer depth will help the computational time.
VERTICAL_RESOLUTION	Vertical resolution (units: meters) used to calculate the vertical mixing criteria values in the PWP model.
BULK_RICHARDSON	Bulk Richardson number used in the PWP model. The default value is 0.65.
GRADIENT_RICHARDSON	Gradient Richardson number used in the PWP model. The default value is 0.25.

27. EnKF Assimilation Setup

```
&NML_ENKF
 ENKF_ON = F
 ENKF_START_DATE = RRK ASSIMILATION START AND END TIME
 ENKF_END_DATE   = For an idealized case specify 'seconds=(flt)','days=(flt)', or 'cycles=(int)'
 ENKF_ASSIM_INTERVAL = A length of time: 'seconds= ','days= ', or 'cycles= '
 ENKF_NOBSMAX   = -1
 ENKF_NENS  = -1
 ENKF_CINF  =  1.0000000E+20
 EKINT_START = 0
 EL_ASSIM = F
 EL_OBS  = F
 UV_ASSIM = F
 UV_OBS  = F
 T_ASSIM = F
 T_OBS  = F
 S_ASSIM = F
 S_OBS  = F
 ENKF_LOCALIZED  = F
 ENKF_METHOD  = 1
 MODE   = 11
 OBSERR_EL  =  1.0000000E-03
 OBSERR_UV =  1.0000000E-03
 OBSERR_T  =  9.9999998E-03
 OBSERR_S  =  9.9999998E-03
 LOCAL_DISK = F
```

ENKF_ON	Controller for turning on/off the EnKF data assimilation. "T" is on and "F" is off.
ENKF_START_DATE	Assimilation start data and time. The format is the same as "START_DATE".
ENKF_END_DATE	Assimilation end data and time. The format is the same as "START_DATE".
ENKF_ASSIM_INTERVAL	The time interval for the assimilation cycle. Three options: 1) "second=", 2) "days=" and 3) "cycle=".
ENKF_NOBSMAX	Maximum number of observations. Note: for the vertical profile at a measurement site, each sampling point is treated as an observation number.
ENKF_NENS	Ensemble size number.
ENKF_CINF	The influence scale of assimilation, which is defined as a radius value in meters. It is implemented for the Cartesian coordinate system. No application has been done for the spherical coordinate system yet. A check is recommended when it is applied for the spherical coordinate system.
EKINT_START	Void parameter. It will be removed.
EL_ASSIM	Controller for the EnKF data assimilation for surface elevation. "T" is on and "F" is off.
EL_OBS	Availability index of surface elevation observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
UV_ASSIM	Controller for the EnKF data assimilation for water velocity. "T" is on and "F" is off.
UV_OBS	Availability index of velocity observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).

T_ASSIM	Controller for the EnKF data assimilation for water temperature. "T" is on and "F" is off.
T_OBS	Availability index of water temperature observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
S_ASSIM	Controller for the EnKF data assimilation for water salinity. "T" is on and "F" is off.
S_OBS	Availability index of water salinity observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
ENKF_LOCALIZED	Controller for the localization. "T" is on and "F" is off. When "T" is selected, ENKF_CINF needs to be defined. If NEKF_CINF is not defined, then the search radius will cover the entire computational domain.
ENKF_METHOD	Choice of the EnKF assimilation types: 1=EnKF ; 2= EnTKF ; 3 =SEIK.
MODE	Selection of the filter methods. This parameter consists of two digits. The first digit is the choice of standard and square root Kalman Filters: "1"-the traditional EnKF "2"-the square root EnKF The second digit is the pseudo inversion option: "1"-eigen value pseudo inversion; "2"- singular vector decomposition with a fixed observational error covariance "3" –singular vector decomposition with a low dimensional representation of observational error covariance.

OBSERR_EL	An estimated observational error of surface elevation, with the same unit as the observational data.
OBSERR_UV	An estimated observational error of velocity, with the same units as the observational data.
OBSERR_T	An estimated observational error of temperature, with the same unit as the observational data.
OBSERR_S	An estimated observational error of salinity with the same unit as the observational data.
LOCAL_DISK	This parameter is designed to control the output to the local disk where the individual computer node is. In the current FVCOM version, this parameter has no function. Please select "F" at all times.

27. RRKF Assimilation Setup

```
&NML_RRKF
RRK_ON = T
REF_START_DATE = RRK REFERENCE  START  TIME FOR EOF CALCULATION"
REF_END_DATE   = For an idealized case specify 'seconds=(flt)','days=(flt)', or 'cycles=(int)'
RRK_START_DATE = RRKF ASSIMILATION START TIME
RRK_END_DATE   = For an idealized case specify 'seconds=(flt)','days=(flt)', or 'cycles=(int)'
RRK_ASSIM_INTERVAL = 'seconds=3600.0' ! A length of time: 'seconds= ','days= ', or 'cycles= '
RRK_NOBSMAX = 50
RRK_NEOF = 8
RRK_PSIZE = 0.05
RRK_PSCALE = 0.05
RRK_RSCALE = 0.001
EL_ASSIM = F
EL_OBS = F
UV_ASSIM = F
UV_OBS = F
T_ASSIM = F
T_OBS = F
S_ASSIM = T
S_OBS = T
LOCAL_DISK = F
```

RRKF_ON	Controller for turning on/off the RRKF data assimilation. "T" is on and "F" is off.
REF_START_DATE	Reference starting time for the EOF calculation. The format is the same as in "START_DATE".

REF_END_DATE	Reference ending time for the EOF calculation. The format is the same as in "START_DATE".
RRK_START_DATE	Assimilation starting data and time. The format is the same as in "START_DATE".
RRK_END_DATE	Assimilation ending data and time. The format is the same as in "START_DATE".
RRK_ASSIM_INTERVAL	The time interval of the assimilation cycle. Three options: 1) "second=", 2) "days=" and 3) "cycle=".
RRK_NOBSMAX	Maximum number of observations. Note: for the vertical profile at a measurement site, each sampling point is treated as an observation number.
RRK_NEOF	Specified number of dominant EOF modes that will be used in the RRKF calculation.
RRK_PSIZE	The magnitude of the perturbation to each EOF mode in calculating the matrix for a quasi-linear theory.
RRK_PSCALE	Scaling parameter to set the pseudo model error proportional to the covariance of the temporal variability from a model run without assimilation.
RRK_RSCALE	Scaling parameter to set the observational error proportional to the covariance of the temporal variability from a model run without assimilation
EL_ASSIM	Controller for the RRKF data assimilation for surface elevation. "T" is on and "F" is off.
EL_OBS	Availability index of surface elevation observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).

UV_ASSIM	Controller for the RRKF data assimilation for water velocity. "T" is on and "F" is off.
UV_OBS	Availability index of velocity observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
T_ASSIM	Controller for the RRKF data assimilation for water temperature. "T" is on and "F" is off.
T_OBS	Availability index of water temperature observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
S_ASSIM	Controller for the RRKF data assimilation for water salinity. "T" is on and "F" is off.
S_OBS	Availability index of water salinity observations. "T" is true (there are observations included in the assimilation) and "F" is false (no observations available).
LOCAL_DISK	This parameter is designed to control the output to the local disk where individual computer node is. In the current FVCOM version, this parameter has no function. Please select "F" at all times.

28. SST Assimilation Setup

```
&NML_SST_ASSIMILATION
 SST_ASSIM     = F
 SST_ASSIM_FILE = example_sst.nc
 SST_RADIUS     =  0.0000000E+00
 SST_WEIGHT_MAX =  0.0000000E+00
 SST_TIMESCALE  =  0.0000000E+00
 SST_TIME_WINDOW =  0.0000000E+00
 SST_N_PER_INTERVAL  =  0
```

SST_ASSIM	Controller for turning on/off the SST assimilation. "T" is on and "F" is off. This assimilation is used without availability of the SST data over the entire computational domain.
SST_ASSIM_FILE	The filename of the SST observational data input file in an ASCII format.
SST_RADIUS	Interpolation search radius in meters.
SST_WEIGHT_MAX	Nudging Gama weight coefficient. Example: 0.5
SST_TIMESCALE	Nudging Alpha weight coefficient. Example: 3.0E-3.
SST_TIME_WINDOW	Nudging time window in seconds. Example: 1800 (s).
SST_N_PER_INTERVAL	Daily mean SST assimilation interval. Example: 24.0 hours.

29. SST Grid Assimilation Setup

```
&NML_SSTGRD_ASSIMILATION
 SSTGRD_ASSIM   = F
 SSTGRD_ASSIM_FILE      = example_split_sstgrd.nc
 SSTGRD_WEIGHT_MAX     =  0.0000000E+00
 SSTGRD_TIMESCALE      =  0.0000000E+00
 SSTGRD_TIME_WINDOW     =  0.0000000E+00
 SSTGRD_N_PER_INTERVAL  =        0
```

SSTGRD_ASSIM	Controller for the SST grid assimilation. "T" is on and "F" is off. The difference of the SST grid assimilation from the SST assimilation is that this assimilation requires all observational data pre-interpolated onto nodes.
SSTGRD_ASSIM_FILE	The filename of the observational SST data input file at node grid points. This file requires a NetCDF format. Example: casename_sstgrd.nc.
SSTGRD_WEIGHT_MAX	Nudging Gama weight coefficient. Example: 0.5
SSTGRD_TIMESCALE	Nudging Alpha weight coefficient. Example: 3.0E-3.
SSTGRD_TIME_WINDOW	Nudging time window in seconds. Example: 1800 (s).

SSTGRD_N_PER_INTERVAL	Daily mean SST assimilation interval. Example: 24.0 hours.

30. SSH Grid Assimilation Setup

```
&NML_SSHGRD_ASSIMILATION
 SSHGRD_ASSIM   = F
 SSHGRD_ASSIM_FILE      = example_split_sshgrd.nc
 SSHGRD_WEIGHT_MAX     = 0.0000000E+00
 SSHGRD_TIMESCALE      = 0.0000000E+00
 SSHGRD_TIME_WINDOW     = 0.0000000E+00
 SSHGRD_N_PER_INTERVAL  = 0
```

SSHGRD_ASSIM	Controller for turning on/off the SSH grid assimilation. "T' is on and "F" is off. Similarly to the SSTGRD_ASSIMILATION, this SSH assimilation requires the input data at node grid points.
SSHGRD_ASSIM_FILE	Filename of the observational SSH data input file in a NetCDF format. Example: "casename_sshgrd.nc" All data must be given at node grid points.
SSHGRD_WEIGHT_MAX	Nudging Gama weight coefficient. Example: 0.5
SSHGRD_TIMESCALE	Nudging Alpha weight coefficient. Example: 3.0E-3.
SSHGRD_TIME_WINDOW	Nudging time window in seconds. Example: 1800 (s).
SSHGRD_N_PER_INTERVAL	Daily mean SST assimilation interval. Example: 24.0 hours.

31. TS Grid Assimilation Setup

```
&NML_TSGRD_ASSIMILATION
 TSGRD_ASSIM   = F
 TSGRD_ASSIM_FILE       = example_split_tsgrd.nc
 TSGRD_WEIGHT_MAX      = 0.0000000E+00
 TSGRD_TIMESCALE =  0.0000000E+00
 TSGRD_TIME_WINDOW      = 0.0000000E+00
 TSGRD_N_PER_INTERVAL   = 0
```

TSGRD_ASSIM	Controller for turning on/off the T/S grid assimilation. This assimilation requires that the observational T/S data are pre-interpolated to the node grid points in the horizontal and standard levels in the vertical. This operation is usually done for restoring of the T/S field (monthly or annually) for a spin up model run experiment. "T" is on and "F" is off.
TSGRD_ASSIM_FILE	The filename of the observational T/S grid input data file in a NetCDF format.
TSGRD_WEIGHT_MAX	Nudging Gama weight coefficient. Example: 0.5
TSGRD_TIMESCALE	Nudging Alpha weight coefficient. Example: 3.0E-3.
TSGRD_TIME_WINDOW	Nudging time window in seconds. Example: 1800 (s).
TSGRD_N_PER_INTERVAL	Daily mean SST assimilation interval. Example: 24.0 hours.

32. Current Nudging Assimilation Setup

```
&NML_CUR_NGASSIMILATION
 CUR_NGASSIM    = F
 CUR_NGASSIM_FILE      = example_split_cur
 CUR_NG_RADIUS  = 0.0000000E+00
 CUR_GAMA      = 0.0000000E+00
 CUR_GALPHA     = 0.0000000E+00
 CUR_NG_ASTIME_WINDOW   = 0.0000000E+00
```

CUR_NGASSIM	Controller for the current nudging assimilation. "T" is on and "F" is off.
CUR_NGASSIM_FILE	The filename of the observational current data input file in an ASCII format. Example: casename_cur.dat.
CUR_NG_RADIUS	The interpolation search radius in meters. Example: 1000.0 (m).
CUR_GAMA	Nudging Gama weight coefficient. Example: 1.0.
CUR_GALPHA	Nudging Alpha weight coefficient. Example: 8.3E-

	3.
CUR_NG_ASTIME_WINDOW	Nudging time window for the current assimilation in seconds. Example: 1800 (s).

33. Current OI Assimilation Setup

&NML_CUR_OIASSIMILATION
 CUR_OIASSIM   = F
 CUR_OIASSIM_FILE   = example_split_cur
 CUR_OI_RADIUS = 0.0000000E+00
 CUR_OIGALPHA = 0.0000000E+00
 CUR_OI_ASTIME_WINDOW = 0.0000000E+00
 CUR_N_INFLU   = 0
 CUR_NSTEP_OI   = 0

CUR_OIASSIM	Controller for the OI current assimilation. "T" is on and "F" is off.
CUR_OIASSIM_FILE	The filename of the observational current input file in an ASCII format. Example: casename_cur.dat.
CUR_OI_RADIUS	Interpolation search radius in meters. Example: 1000.0 (m).
CUR_OIGALPHA	OI Alpha weight coefficient. Example: 8.3E-3.
CUR_OI_ASTIME_WINDOW	OI time window for current assimilation in seconds. Example: 1800 (s).
CUR_N_INFLU	Number of influential observations. Example: 1.0.
CUR_NSTEP_OI	Time step interval for the current assimilation. Example: 10.

33. TS Nudging Assimilation Setup

&NML_TS_NGASSIMILATION
 TS_NGASSIM   = F
 TS_NGASSIM_FILE = example_split_ts
 TS_NG_RADIUS   = 0.0000000E+00
 TS_GAMA = 0.0000000E+00
 TS_GALPHA   = 0.0000000E+00
 TS_NG_ASTIME_WINDOW   = 0.0000000E+00

TS_NGASSIM	Controller for the T/S nudging data assimilation. "T"

	is on and "F" is off.
TS_NGASSIM_FILE	The filename of the observational T/S input file in an ASCII format.
TS_NG_RADIUS	Interpolation search radius in meters. Example: 1000.0 (m).
TS_GAMA	Nudging Gama weight coefficient. Example: 1.0.
TS_GALPHA	Nudging Alpha weight coefficient. Example: 8.3E-3.
TS_NG_ASTIME_WINDOW	Time interval window for T/S assimilation in seconds. Example: 1800 (s).

34. TS OI Assimilation Setup

```
&NML_TS_OIASSIMILATION
 TS_OIASSIM    = F
 TS_OIASSIM_FILE = example_split_ts
 TS_OI_RADIUS   = 0.0000000E+00
 TS_OIGALPHA    = 0.0000000E+00
 TS_OI_ASTIME_WINDOW    = 0.0000000E+00
 TS_MAX_LAYER   = 0
 TS_N_INFLU  = 0
 TS_NSTEP_OI = 0
```

TS_OIASSIM	Controller for the OI T/S data assimilation. "T' is on and "F" is off.
TS_OIASSIM_FILE	The filename of observational T/S input file in an ASCII format. Example: casename_ts.dat.
TS_OI_RADIUS	Interpolation search radius in meters. Example: 1000.0 (m).
TS_OIGALPHA	OI Alpha weight coefficient. Example: 8.3E-3.
TS_OI_ASTIME_WINDOW	Time interval window for OI T/S assimilation.
TS_MAX_LAYER	Maximum numbers of data in the vertical from any surrounding observational sites that will be considered in the assimilation. Example: 20.
TS_N_INFLU	Number of influential observations. Example: 1.0
TS_NSTEP_OI	The time step interval for the OI T/S assimilation. Example: 10.

18.2 FVCOM Input Files

FVCOM input files must be placed in the directory named "input" by the variable INPDIR in the runtime control parameter file described in section 18.1. All input files are prefixed by the string "casename" referring to the application description string chosen by the user. Users can use different filenames, as long as the filenames are the same as those listed in the *casename_run.nml*. A description of the input files and their primary data is provided below.

18.2.1 ASCII Format Input Files

1) **Casename_cor.dat**	The latitudes of triangular nodes that are used to calculate Coriolis parameter. Format: ASCII.
2) **Casename_dep.dat**	The water depth at all node points. Format: ASCII.
3) **Casename_grd.dat**	Triangular meshes (numbers of individual triangle identification and its three nodes and the *x* and *y* locations of individual node points). Format: ASCII.
4) **Casename_lsf.dat**	Identification number of open boundary nodes where a frictional geostrophic inflow correction is made. This input file is usually used in a near-shore upstream open boundary where the inflow can be parameterized by the wind stress. This condition was added by L. Pringle (UNH) to add the inflow on the Scotian Shelf in the Gulf of Maine/Georges Bank model. Format: ASCII.

5) **Casename_obc.dat**	The node numbers of open boundaries. Format: ASCII file.
6) **Casename_sigma.dat**	The vertical coordinate setup file. Format: ASCII.
7) **Casename_spg.dat**	Parameters for a sponge layer for damping at the open boundaries. Format: ASCII.
8) **Casename_node_nest.dat**	The node numbers for large domain nesting output. Format: ASCII.
9) **Casename_node_nest_wave.dat**	The node numbers for large domain wave nesting output. Format: ASCII.
10) **Casename_station.dat**	Node or cell numbers for the time series output at stations (selected nodes or cell). This file includes node (or cell) numbers, locations, water depths and station names. Format: ASCII.
11) **Casename_ts.xy**	The input file used for the T/S assimilation, which includes the total number and locations (*x, y, z*) of observations and also Cell IDs where the observations were located. Format: ASCII
12) **Casename_ts.dat**	The input file of observed water temperature and salinity at measurement times. This file includes 1) the measurement time (Julian day), 2) temperature, and 3) salinity. Format: ASCII.
13) **Casename_cur.xy**	The input file used for the velocity assimilation, which includes the total number and locations (*x, y, z*) of observations and also

	Cell IDs where the observations were located. Format: ASCII.
14) **Casename_cur.dat**	The input file of observed velocity at measurement times. This file includes 1) the measurement time (Julian day), 2) u, and 3) v. Format: ASCII.

18.2.2 NetCDF Format Input Files

1) **Casename_julian_obc.nc**	Elevations at the open boundary (for the case with the Julian time or real o'clock time). Format: NetCDF.
2) **Casename_non_julian_obc.nc**	Tidal amplitudes and phases at the open boundary (for the case with the non-Julian time). Format: NetCDF.
3) **Casename_wnd.nc**	The real-time field of wind velocity or wind stress used for external and internal modes (split solver) and internal modes (semi implicit solver). In FVCOM v3.1.6 or up, this file also can include surface heat flux, shortwave irradiance, air pressure, precipitation/evaporation, and other meteorological forcing data. Format: NetCDF.
4) **Casename_restart.nc**	The restart file used to restart the FVCOM run from a restart time with the model output values at that time or initial values. Format: NetCDF.
5) **Casename_tsobc.nc**	The time series of the temperature and salinity

	used for nudging on open boundary nodes. This is an input file that is usually constructed from observational data or output from larger domain FVCOM or other models Format: NetCDF.
6) **Casename_river.nc**	River discharge data including number of rivers, discharge volume, discharge temperature, and discharge salinity. Format: NetCDF.
7) **Casename_node_nest.nc**	The nesting boundary file output from a larger domain and used as an input file to specify the nesting boundary condition for a small domain. Format: NetCDF.
8) **Casename_node_nest_wave.nc**	The nesting boundary file output from a larger domain SWAVE model run and used as an input file to specify the nesting boundary condition for a small domain SWAVE. Format: NetCDF.
9) **Casename_sst.nc**	The sea surface temperature (SST) input file used for the SST data assimilation. Format: NetCDF.
10) **Casename_hvc.nc**	Coefficients used to calculate the horizontal diffusion in momentum and scalar equations. Format: NetCDF.
11) **Casename_brf.nc**	Bottom roughness coefficients. Format: NetCDF.
12) **Casename_lag_start.nc**	Initial positions of particles to be tracked using the Lagrangian tracking module. Format: NetCDF.
13) **Casename_lag_restart.nc**	The restart file used for the particle tracking

	at a re-start time. The contents are the same as the initial position file described in "Casename_lag_start.nc". Format: NetCDF.
14) **Casename_grndwtr.nc**	The bottom groundwater input values including number of sources, time, and discharge rate, etc. Format: NetCDF.
15) **Casename_hfx.nc**	The heat flux parameters that include the air temperature, relative humidity, surface pressure, longwave radiation and shortwave irradiance). These data are used to recalculate sensible and latent heat flux and hence net heat flux. Format: NetCDF.
16) **Casename_ice_forcing_nc**	The input file for the ice model, which includes variables of specific humidity, air temperature (could be available in the *casename_hfx.nc*) and cloud cover. Format: NetCDF.
17) **Casename_air_press.nc**	The input file of the surface air pressure (unit: Pa). Format: NetCDF.
18) **Casename_pre_evap.nc**	The input file of precipitation and evaporation (unit: m/s). Format: NetCDF.
19) **Casename_ssh.nc**	The input file used for the daily SSH assimilation, which includes absolute dynamics topography (not the mean sea level anomaly that is used usually in other models). Format: NetCDF.

20) **Casename_PTS.nc**	The input file used for the T/S monthly restoration, which includes monthly averaged temperature and salinity). Format: NetCDF.
21) Casename_mf.nc	The input file used to specify the mean flow on the open boundary. Format: NetCDF.

18.3. Input Files Required for Biological and Sediment Modules

1. GEM

The setup of the GEM run is straightforward. To run the online GEM, one must edit the "*biomodel.in*" to select the desired code modules and parameters. An example is included in the FVCOM source code package. In addition to physical parameters and input files, the biological model run is controlled by initial and boundary conditions. The boundary conditions are specified. The boundary condition input file is '*Casename_bio_obc.nc*". Unlike physical fields, biological measurements are usually made at a scatter of points that are not sufficient to build an initial field for the modeling effort. For this reason, the initial fields of the NPZ, NPZD, NPZDB and water quality models are specified by either the steady state solutions of the governing equations or 1-D (vertical) profiles derived from available observations. FVCOM v3.1.6 or up has generalized the biological input files based on variables. For example, a simple single variable NPZ model requires three input files: NUTRIENT_INI_1.dat; PHYPOLANKTON_INI_1.dat; ZOOPLANKTON_INT_1.DAT. For NPZD model which includes two nutrients, two phytoplankton, two zooplankton, and one detritus component, the input files should be

NUTRIENT_INI_1.dat
NUTRIENT_INI_2.dat
PHYTOPLANKTON_INI_1.dat
PHYTOPLANKTON_INI_2.dat
ZOOPLANKTON_INI_1.dat
ZOOPLANKTON_INI_2.dat
DETRITUS_INI_1.dat

2. OFFLINE GEM

The offline GEM has the same code structure as the online GEM. The only difference is that it requires the physical model input files.　After editing "*biomodel.in*", one needs to specify 1) initial conditions, 2) physical forcing; 3) a NetCDF FVCOM physical model output field (it can be multiple files depending on the FVCOM output setup); 4) boundary conditions, and 5) river inputs, etc.　The initial, boundary and river input files are all in an ASCII format.

3. OFFLINE UG-RCA

Running offline UG-RCA requires physical field files output from FVCOM and forcing and initial fields from either observations or others.　How many input files are required to run this model depends on the case users consider.　Here we gave an example of input files we used to drive UG-RCA in Massachusetts Bay for the 2009 simulation run.　These files include:

bin2009_fvcom.ic	The UG-RCA initial fields, which are from the output of the previous year simulation result.
atmos2009_fvcom_load	The atmospheric loading.
bcf2009_fvcom.2wk	Boundary forcing that are specified using bi-weekly survey data.
model_fcvom.fl2009	Nutrient loading from rivers.
model_fvcom_pcv.2009	Shortwave radiation (output from FVCOM).
nps_fvcom.2009	Non-point sources of nutrient flux.
ps_fvcom.2009.wk	Point sources of nutrient flux (e.g., MWRA pipe discharge).
sed2009_fvcom.ic	The initial sediment distribution, which is output from the previous year simulation result.

4. UG_CE-QUAL-ICM

Converting CE-QUAL-ICM into an unstructured-grid finite-volume model was approached using the same framework of FVCOM.　Running this model requires the physical fields from the output of FVCOM, biological forcing, initial and boundary input

files. An example is given in the directory where the UG_CE_QUAL_ICM is located. That experiment was made by PNL scientists.

5. SEDIMENT MODEL

The input files for the sediment model is described in detail in the *generic_sediment.inp*, which are included in the FVCOM source code. Most of them are parameters used to control the sediment model run.

18.4 ASCII Input File Formats for Primary Input Files

1. Casename_cor.dat

Node number = xxx

x	*y*	Lat
(meter)	(meter)	(degree)
(or long.)	(or lat.)	

This is a data array with three columns. (*x*, *y*) and Lat are the location and latitude of individual node points on each triangular mesh. Total rows of this array are equal to the total number of node points. Header information of the total node number is required in this input file. In a spherical coordinate, *x* and *y* are longitude and latitude in degree.

Example:

```
Node Number = 48860
 1699880.00   420500.00   46.145951
 1705180.00   414890.00   46.090111
 1709910.00   409690.00   46.038550
 1715778.00   403368.30   45.975706
 1722603.00   395907.00   45.901624
 1730039.00   387611.30   45.819404
     :            :            :
     :            :            :
```

2. Casename_dep.dat

Node Number = xxxx

x	*y*	*d*
(meter)	(meter)	(meter)
(or long.)	(or lat.)	

This is a data array with three columns. (*x*, *y*) and *d* are the location and water depth of individual node points on each triangular mesh. Total rows of this array are equal to the total number of node points. The depth in FVCOM is specified at the node point. Header information of the total node number is required in this input file.

For estuaries, the water depth was usually measured at

the lowest water level. Therefore, the adjustment depth must be specified.

Example:

```
Node Number = 48860
  1699880.00   420500.00     9.166
  1705180.00   414890.00     5.000
  1709910.00   409690.00     5.731
  1715778.00   403368.30    32.689
  1722603.00   395907.00    75.348
  1730039.00   387611.30   118.228
  1737004.00   380188.10   177.054
       :            :            :
       :            :            :
       :            :            :
```

3. Casename_grd.dat

Node Number = xxxx Cell Number = xxxx	The grid input file consists of two parts: 1) integral numbers identifying elements and nodes and (2) the x and y locations of individual nodes.

E N_1 N_2 N_3 Type
 	
  :   :   :   :   1
 	
  :   :   :   :   1
 	
  :   :   :   :   1
	Column E: Identify integral number of individual element I (SMS program will list the elements from 1, 2… to N, but FVCOM can identify a random order of the element number file).
	
	Column N_1-N_3: Identify integral numbers of the nodes of the element I listed in the same row.
	
	Column Type: The element type information which automatically comes out from SMS grid generation program. This information is not needed for FVCOM. For user's created grid file, this column can be removed.

N   X   Y
     (meter)   (meter)
    (or. long.)   (or. lat.)
  :   :   :
  :   :   :
  :   :   :
	Column N: Identify integral number of individual node.
	
	Columns X and Y: The x and y locations of the node I shown in the same row.
	
	Note 1: In the grid file created by SMS, there is a fourth column in this part. That information is never used in FVCOM. For user's created grid file, this column can be removed.
	
	Note 2: When integrating FVCOM in Spherical Coordinates, X and Y are Latitude and Longitude respectively.

Example:

```
Node Number =  48860
Cell Number =  91258
      1    96    95    1    1
      2    95     2    1    1
      3     3     2   95    1
      4    95    97    3    1
      5    97    98    3    1
      6     4     3   98    1
.
.
   1  1699880.00   420500.00
   2  1705180.00   414890.00
   3  1709910.00   409690.00
   4  1715778.00   403368.30
   5  1722603.00   395907.00
   6  1730039.00   387611.30
.
.
```

4. Casename_lsf.dat

Longshore Flow Node Number =N_OBS

N_OBS: Total number of the open boundary node points.

N   GL_NODE   GEO   WDF

N: User defined index number from 1 to N_OBS;

x   x   x   x

:   :   :   :

GL_NODE: The identification number of individual open boundary node points.

GEO: Thermal wind flow adjusting scaling in a range of [0 1].

WDF: Wind driven flow adjusting scaling in a range of [0 1].

Example:

```
Longshore Flow Node Number = 33
# NODES MUST BE LISTED IN THE OFFSHORE DIRECTION
#    GL_NODE GEO    WDF
1        1      1.0      0.5
2        2      1.0      0.25
3        3      1.0      0.0
4        4      1.0      0.0
5        5      1.0      0.0
6        6      1.0      0.0
7        7      1.0      0.0
```

8   8   1.0   0.0
.
.

5. Casename_obc.dat

OBC Node Number = xxxx	No.: counting number of the open boundary node from 1 to OBC Node_Number.

No.	Node_ID	Node_Type
1	x	x
2	x	x
.	.	.
.	.	.
NOB	x	x

Node_ID: The node ID on the open boundary.

Node Type: Specifies method of setting surface elevation. Use 0 for prescribed elevation (Julian/non-Julian). Use 1 for a radiation condition.

Example:

```
OBC Node Number = 10
    1   1   1
    2   2   1
    3   3   1
    4   4   1
    5   5   1
    6   6   1
    7   7   1
    8   8   1
    9   9   1
   10  10   1
```

6. Casename_sigma.dat

FVCOM includes three types of terrain-following coordinates: 1) the sigma coordinate, 2) general vertical coordinate, and 3) hybrid coordinate. In all coordinates, the first line in the *casename_sigma.dat* is the total number of vertical levels. In FVCOM v3.1.6 or up, we define the vertical coordinate by "SIGMA COORDINATE TYPE". The sigma coordinate can be classified as "uniform" or "geometric". The general vertical coordinate is classified as "tanh" and hybrid coordinate is defined as "generalized".

For the sigma coordinate, the sigma level is specified as

$$\sigma(k) = [(k-1)/(kb-1)]^{P-SIGMA}$$

where *kb* is the total number of the sigma level and *P_SIGMA* can be any real number. For example,

P_SIGMA = 1: Uniform sigma layers;

P_SIGMA = 2: Layer satisfying a parabolic function with high vertical resolution near the surface and bottom.

For the general vertical coordinate, the vertical level is determined by the equation given as

$$z(k) = \frac{\tanh(DU + DL)((KB-1-K)/(KB-1)) - DL) + \tanh(DL)}{\tanh(DU) + \tanh(DL)} - 1$$

where DU is the upper depth boundary from the surface, up to which the coordinates are parallel with uniform thickness; and DL is the lower depth boundary from the bottom, down to which the coordinates are parallel with uniform thickness.

For the hybrid coordinate, a general vertical coordinate (or s-coordinate) is specified in the deep ocean region and a sigma coordinate is specified in the shallow water region. These two coordinates have a transition at a depth defined as the minimum water depth (HMIN1).

Example for the hybrid coordinate:

```
NUMBER OF SIGMA LEVELS = 41                  % Total vertical levels
SIGMA COORDINATE TYPE = GENERALIZED     %Hybrid coordinate
DU = 25.0                   % Upper water boundary thickness (meters)
DL = 25.0                    % Lower water boundary thickness (meters)
MIN CONSTANT DEPTH = 200.0   %The transition depth of the hybrid coordinate
KU = 5                  %Layer number in the water column of DU
KL = 5                  %Layer number in the water column of DL
ZKU = 5. 5. 5. 5. 5.      %Thickness of each layer defined by KU (meters).
ZKL = 5. 5. 5. 5. 5.      %Thickness of each layer defined by KL (meters).
```

Example for the uniform sigma coordinate:

```
NUMBER OF SIGMA LEVELS = 41                %Total vertical levels
SIGMA COORDINATE TYPE = UNIFORM     %Uniform sigma level (P_SIGMA = 1)
```

Example for the high-order sigma coordinate:

```
NUMBER OF SIGMA LEVELS = 41                 %Total vertical levels
SIGMA COORDINATE TYPE = GEOMETRIC   %Geometric sigma level
SIGMA POWER= 2.0                           %P_SIGMA = 2
```

7. Casename_spg.dat

Sponge Node Number =N_sp

Node_ID (1), R(1), F_f (1)

Node_ID (2), R(2), F_f (2)

………

………

Node_ID(N_sp), R(N_sp), F_f(N_sp)

N_sp: Total number of outer-side boundary nodes at which the friction is added.

Node_ID(i): The ith sponge center node ID

R(i): The ith sponge layer's affecting influence radius.

F_f(i): Damping coefficient of the ith sponge layer.

Example:

```
Sponge Node Number = 5
  1   20000   0.001
  2   20000   0.001
  3   20000   0.001
  4   20000   0.001
  5   20000   0.001
```

8. Casename_node_nest.dat

Node_Nest_ Number = xxxx

No.   Node_ID Node_Type

1   x   x

2   x   x

.   .   .

.   .   .

NOB   x   x

Node_Nest_Number: Total number of nodes on the nesting boundary.

No.: Counting number of the open boundary node from 1 to OBC Node_Number.

Node_ID: Node ID on the nesting boundary.

Node Type: Always keep an integer of "1". Not used in the program, but required to read in.

Example:

```
Node_Nest Number = 5
      1     34760      1
      2     34268      1
      3     34269      1
      4     33767      1
      5     33259      1
```

9. Casename_node_nest_wave.dat

The data content and structure of this wave nesting input file are the same as in the *casename_node_nest.dat*.

Example:

```
Node_Nest Number = 5
      1    34760       1
      2    34268       1
      3    34269       1
      4    33767       1
      5    33259       1
```

10. Casename_station.dat

No.	X Long (or meters)	Y Lat (or meters)	Node (or Cell) ID	Depth (m)	Station name
x	x	x	x	x	x

Column 1: Counting number from 1 to N where N is the total station number.

Columns 2-3: Location of each station. For the Cartesian coordinates, they are meters.

For the spherical coordinate, they are longitude and latitude.

Column 4: Node or cell ID numbers. The total station number need to be specified in the namelist file, in which node or cell needs to be selected.

Column 5: Station information.

Example:

```
No    X          Y        Node   Depth (m)     Station Name
1  -85.66666   30.15167    170    27.49       '8729108 Panama City, FL'
2  -86.49333   30.50333   50759   2.29        '8729501 Valparaiso, FL     '
3  -87.21167   30.40333   59185   7.66        '8729840 Pensacola, FL      '
4  -87.42834   30.38667   39686   4.41        '8729941 Blue Angles PK, FL '
5  -87.68333   30.27833   40673   5.50        '8731439 Gulf Shores ICWW, AL'
```

11. Casename_ts.xy

```
 N_ASSIM_TS
 No.  X   Y   DEPTH  N_LAYERS  SITA  N_CELL
ODEPTH(1)
……..
ODEPTH(N_LAYERS)
```

Here:

N_ASSIM_TS: number of temperature and salinity observations;

No.; Observational station counting number;

X and Y: The location of observations;

DEPTH: Local depth at observational stations (meters).

N_LAYERS: number of measurements in the vertical at each station;

SITA: Angle of the observation station to the local isobaths;

N_Cell: The cell ID where the observational stations are located;

ODEPTH(N_LAYERS): Measurement depths (meters).

Example:

```
 2
 1   1034714.468   -366785.640  300.0   2  0.0   1254
 0.00
 5.00
 2  1570883.557      9949.789  300.0   3  0.0     929
 0.00
 5.00
10.00
```

12. Casename_ts.dat

```
Nsite  N_TIMES
ODAYS(1)   T1 S1 T2 S2 …………………….TN SN
ODAYS(2)   T1 S1 T2 S2 …………………….TN SN
  …          .   .   .  .       …………………....   .


ODAYS(N_times)  T1 S1 T2 S2 ………………..TN SN
```

Here:

Nsite: Number of observations;

N_TIME: Number of measurements in time;

ODAYS: Julian time when the measurements were made;

T1, S1: Temperature and salinity at ODEPTH(1);

T2, S2: Temperature and salinity at ODEPTH(2);

  .. ..

TN, SN: Temperature and salinity at ODEPTH(N).

Example:

```
 1   2
```

```
54952.000000  -99.90  -99.90  18.85  36.07
54952.100000  19.00  36.00  18.85  36.07
 2   3
54952.000000  -99.90  -99.90  18.85  36.07
54952.100000  19.00  36.00  18.85  36.07
54953.100000  19.20  35.90  18.95  36.00
```

13 Casename_cur.xy

```
N_ASSIM_CUR
No.  X   Y   DEPTH  N_LAYERS  SITA  N_CELL
ODEPTH(1)
……
ODEPTH(N_LAYERS)
```

Here:

N_ASSIM_CUR: Number of current observations

No.; Observational station counting number;

X and Y: The location of observations;

DEPTH: Local depth at observational stations (meters).

N_LAYERS: number of measurements in the vertical at each station;

SITA: Angle of the observation station to the local isobaths;

N_Cell: The cell ID where the observational stations are located;

ODEPTH(N_LAYERS): Measurement depths (meters).

Example:

```
  3
1    331.7948  5.4952  2000.0  2  0.0   579
0.00
5.00
2   333.3704  4.7476  2000.0  2  0.0    871
0.00
5.00
3   330.4846  2.3592  2000.0   3  0.0  3658
0.00
5.00
10.00
```

14. Casename_cur.dat

```
No.  N_TIME
ODAYS(1)   U(1) V(1) U(2) V(2) ……………..U(N_LAYERS)  V(N_LAYERS)
   ………………………………..
ODAYS(N_TIME)  U(1) V(1)   U(2) V(2)……. U(N_LAYERS)  V(N_LAYERS)
```

Example:

```
1   2
  54952.625000  -15.85   15.93  -15.85   15.93
  54952.665000  -16.85   14.93  -16.85   14.93
 2   1
  54952.750000  -13.72   20.55  -13.72   20.55
 3   1
  54952.750000   24.61   -7.24   24.61   -7.24   24.61   -7.24
```

18.5 NetCDF Input File Formats for Primary Input Files

The FVCOM development team and users have created Fortran and Matlab programs to create the NetCDF input files. Several Fortran programs are provided in the directory "./FVCOM_source/input/". Matlab toolbox is served via svn on a google project page with website address:

<div align="center">

http://code.google.com/p/fvcom-toolbox

</div>

The Matlab toolbox for pre-processing is included in the FVCOM directory named "Pre-processing", too. However, we strongly recommend that users use the google projects to get the newest version since it is being modified and improved fairly regularly. We welcome users to contribute their own experiences to improve FVCOM pre-processing and post-processing toolbox.

Some examples of the NetCDF input file information are displayed below using "ncdump –h " command.

1. Casename_julian_obc.nc

ncdump -h Casename_julian_obc.nc:

```
netcdf Casename_julian_obc {
dimensions:
    nobc = 94 ;
    time = UNLIMITED ; // (43922 currently)
    DateStrLen = 26 ;
variables:
    int obc_nodes(nobc) ;
        obc_nodes:long_name = "Open Boundary Node Number" ;
        obc_nodes:grid = "obc_grid" ;
    int iint(time) ;
        iint:long_name = "internal mode iteration number" ;
    float time(time) ;
        time:long_name = "time" ;
```

```
            time:units = "days since 1858-11-17 00:00:00" ;
            time:format = "modified julian day (MJD)" ;
            time:time_zone = "UTC" ;
        int Itime(time) ;
            Itime:units = "days since 1858-11-17 00:00:00" ;
            Itime:format = "modified julian day (MJD)" ;
            Itime:time_zone = "UTC" ;
        int Itime2(time) ;
            Itime2:units = "msec since 00:00:00" ;
            Itime2:time_zone = "UTC" ;
        char Times(time, DateStrLen) ;
            Times:time_zone = "UTC" ;
        float elevation(time, nobc) ;
            elevation:long_name = "Open Boundary Elevation" ;
            elevation:units = "meters" ;

// global attributes:
        :type = "FVCOM TIME SERIES ELEVATION FORCING FILE" ;
        :title = "JULIAN FVCOM TIDAL FORCING DATA CREATED FROM OLD FILE TYPE: No
comments found... this is mystery data!" ;
        :history = "FILE CREATED: 20100825T101547.253" ;
}
```

2. Casename_non_julian_obc.nc

ncdump -h Casename_non_julian_obc.nc

```
netcdf Casename_non_julian_obc {
dimensions:
    nobc = 25 ;
    tidal_components = 1 ;
    DateStrLen = 26 ;
variables:
    int obc_nodes(nobc) ;
        obc_nodes:long_name = "Open Boundary Node Number" ;
        obc_nodes:grid = "obc_grid" ;
    float tide_period(tidal_components) ;
        tide_period:long_name = "tide angular period" ;
        tide_period:units = "seconds" ;
    float tide_Eref(nobc) ;
        tide_Eref:long_name = "tidal elevation reference level" ;
        tide_Eref:units = "meters" ;
    float tide_Ephase(tidal_components, nobc) ;
        tide_Ephase:long_name = "tidal elevation phase angle" ;
        tide_Ephase:units = "degrees, time of maximum elevation with respect to chosen time origin" ;
    float tide_Eamp(tidal_components, nobc) ;
        tide_Eamp:long_name = "tidal elevation amplitude" ;
        tide_Eamp:units = "meters" ;
    float equilibrium_tide_Eamp(tidal_components) ;
        equilibrium_tide_Eamp:long_name = "equilibrium tidal elevation amplitude" ;
        equilibrium_tide_Eamp:units = "meters" ;
    float equilibrium_beta_love(tidal_components) ;
        equilibrium_beta_love:formula = "beta=1+klove-hlove" ;
    char equilibrium_tide_type(tidal_components, DateStrLen) ;
        equilibrium_tide_type:long_name = "formula" ;
```

```
            equilibrium_tide_type:units = "beta=1+klove-hlove" ;
       float time_origin ;
            time_origin:long_name = "time" ;
            time_origin:units = "days since 0.0" ;
            time_origin:time_zone = "none" ;

// global attributes:
            :type = "FVCOM SPECTRAL ELEVATION FORCING FILE" ;
            :title = "Spectral forcing data from:tst_el_obc.dat" ;
            :components = "M2" ;
            :history = "FILE CREATED: 2008-04-22T21:40:48Z: UTC" ;
}
```

3. Casename_wnd.nc

ncdump -h Casename_wnd.nc

```
netcdf Casename_wnd {
dimensions:
     nele = 91258 ;
     node = 48860 ;
     three = 3 ;
     time = UNLIMITED ; // (3625 currently)
     DateStrLen = 26 ;
variables:
     int nprocs ;
          nprocs:long_name = "number of processors" ;
     int partition(nele) ;
          partition:long_name = "partition" ;
     float x(node) ;
          x:long_name = "nodal x-coordinate" ;
          x:units = "meters" ;
     float y(node) ;
          y:long_name = "nodal y-coordinate" ;
          y:units = "meters" ;
     float lon(node) ;
          lon:long_name = "nodal longitude" ;
          lon:standard_name = "longitude" ;
          lon:units = "degrees_east" ;
     float lat(node) ;
          lat:long_name = "nodal latitude" ;
          lat:standard_name = "latitude" ;
          lat:units = "degrees_north" ;
     float xc(nele) ;
          xc:long_name = "zonal x-coordinate" ;
          xc:units = "meters" ;
     float yc(nele) ;
          yc:long_name = "zonal y-coordinate" ;
          yc:units = "meters" ;
     float lonc(nele) ;
          lonc:long_name = "zonal longitude" ;
          lonc:standard_name = "longitude" ;
          lonc:units = "degrees_east" ;
     float latc(nele) ;
```

```
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
int nv(three, nele) ;
        nv:long_name = "nodes surrounding element" ;
int iint(time) ;
        iint:long_name = "internal mode iteration number" ;
float time(time) ;
        time:long_name = "time" ;
        time:units = "days since 1858-11-17 00:00:00" ;
        time:format = "modified julian day (MJD)" ;
        time:time_zone = "UTC" ;
int Itime(time) ;
        Itime:units = "days since 1858-11-17 00:00:00" ;
        Itime:format = "modified julian day (MJD)" ;
        Itime:time_zone = "UTC" ;
int Itime2(time) ;
        Itime2:units = "msec since 00:00:00" ;
        Itime2:time_zone = "UTC" ;
char Times(time, DateStrLen) ;
        Times:time_zone = "UTC" ;
char file_date(time, DateStrLen) ;
        file_date:time_zone = "UTC" ;
float short_wave(time, node) ;
        short_wave:long_name = "Short Wave Radiation" ;
        short_wave:units = "W m-2" ;
        short_wave:grid = "fvcom_grid" ;
        short_wave:coordinates = "" ;
        short_wave:type = "data" ;
float net_heat_flux(time, node) ;
        net_heat_flux:long_name = "Surface Net Heat Flux" ;
        net_heat_flux:units = "W m-2" ;
        net_heat_flux:grid = "fvcom_grid" ;
        net_heat_flux:coordinates = "" ;
        net_heat_flux:type = "data" ;
float uwind_stress(time, nele) ;
        uwind_stress:long_name = "Eastward Wind Stress" ;
        uwind_stress:standard_name = "Wind Stress" ;
        uwind_stress:units = "Pa" ;
        uwind_stress:grid = "fvcom_grid" ;
        uwind_stress:type = "data" ;
float vwind_stress(time, nele) ;
        vwind_stress:long_name = "Northward Wind Stress" ;
        vwind_stress:standard_name = "Wind Stress" ;
        vwind_stress:units = "Pa" ;
        vwind_stress:grid = "fvcom_grid" ;
        vwind_stress:type = "data" ;
float U10(time, nele) ;
        U10:long_name = "Eastward Wind Speed" ;
        U10:units = "m/s" ;
        U10:grid = "fvcom_grid" ;
        U10:coordinates = "" ;
        U10:type = "data" ;
float V10(time, nele) ;
        V10:long_name = "Northward Wind Speed" ;
        V10:units = "m/s" ;
```

```
        V10:grid = "fvcom_grid" ;
        V10:coordinates = "" ;
        V10:type = "data" ;
    float precip(time, node) ;
        precip:long_name = "Precipitation" ;
        precip:description = "Precipitation, ocean lose water is negative" ;
        precip:units = "m s-1" ;
        precip:grid = "fvcom_grid" ;
        precip:coordinates = "" ;
        precip:type = "data" ;
    float evap(time, node) ;
        evap:long_name = "Evaporation" ;
        evap:description = "Evaporation, ocean lose water is negative" ;
        evap:units = "m s-1" ;
        evap:grid = "fvcom_grid" ;
        evap:coordinates = "" ;
        evap:type = "data" ;
    float uwind_speed(time, nele) ;
        uwind_speed:long_name = "Eastward Wind Speed" ;
        uwind_speed:standard_name = "Wind Speed" ;
        uwind_speed:units = "m/s" ;
        uwind_speed:grid = "fvcom_grid" ;
        uwind_speed:type = "data" ;
    float vwind_speed(time, nele) ;
        vwind_speed:long_name = "Northward Wind Speed" ;
        vwind_speed:standard_name = "Wind Speed" ;
        vwind_speed:units = "m/s" ;
        vwind_speed:grid = "fvcom_grid" ;
        vwind_speed:type = "data" ;

// global attributes:
        :title = "\'AN FVCOM CASE DESCRIPTION\' - note string must be in \'quotes\'" ;
        :institution = "School for Marine Science and Technology" ;
        :source = "FVCOM grid (unstructured) surface forcing" ;
        :history = "model started at: 25/10/2010   16:56" ;
        :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
        :Conventions = "CF-1.0" ;
        :CoordinateSystem = "Cartesian" ;
        :CoordinateProjection = "init=nad83:1802" ;
}
```

4. Casename_restart.nc

ncdump -h Casename_restart.nc

```
netcdf Casename_restart {
dimensions:
    nele = 91258 ;
    node = 48860 ;
    siglay = 40 ;
    siglev = 41 ;
    three = 3 ;
    time = UNLIMITED ; // (1 currently)
    DateStrLen = 26 ;
    nobc = 94 ;
```

```
    nlsf = 33 ;
variables:
    int nprocs ;/usr/share/doc/HTML/index.html
        nprocs:long_name = "number of processors" ;
    int partition(nele) ;
        partition:long_name = "partition" ;
    float x(node) ;
        x:long_name = "nodal x-coordinate" ;
        x:units = "meters" ;
    float y(node) ;
        y:long_name = "nodal y-coordinate" ;
        y:units = "meters" ;
    float lon(node) ;
        lon:long_name = "nodal longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lat(node) ;
        lat:long_name = "nodal latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    float xc(nele) ;
        xc:long_name = "zonal x-coordinate" ;
        xc:units = "meters" ;
    float yc(nele) ;
        yc:long_name = "zonal y-coordinate" ;
        yc:units = "meters" ;
    float lonc(nele) ;
        lonc:long_name = "zonal longitude" ;
        lonc:standard_name = "longitude" ;
        lonc:units = "degrees_east" ;
    float latc(nele) ;
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
    float siglay(siglay, node) ;
        siglay:long_name = "Sigma Layers" ;
        siglay:standard_name = "ocean_sigma/general_coordinate" ;
        siglay:positive = "up" ;
        siglay:valid_min = -1.f ;
        siglay:valid_max = 0.f ;
        siglay:formula_terms = "sigma: siglay eta: zeta depth: h" ;
    float siglev(siglev, node) ;
        siglev:long_name = "Sigma Levels" ;
        siglev:standard_name = "ocean_sigma/general_coordinate" ;
        siglev:positive = "up" ;
        siglev:valid_min = -1.f ;
        siglev:valid_max = 0.f ;
        siglev:formula_terms = "sigma:siglay eta: zeta depth: h" ;
    float h(node) ;
        h:long_name = "Bathymetry" ;
        h:standard_name = "sea_floor_depth_below_geoid" ;
        h:units = "m" ;
        h:positive = "down" ;
        h:grid = "Bathymetry_Mesh" ;
        h:coordinates = "" ;
        h:type = "data" ;
```

```
int nv(three, nele) ;
    nv:long_name = "nodes surrounding element" ;
int iint(time) ;
    iint:long_name = "internal mode iteration number" ;
float time(time) ;
    time:long_name = "time" ;
    time:units = "days since 1858-11-17 00:00:00" ;
    time:format = "modified julian day (MJD)" ;
    time:time_zone = "UTC" ;
int Itime(time) ;
    Itime:units = "days since 1858-11-17 00:00:00" ;
    Itime:format = "modified julian day (MJD)" ;
    Itime:time_zone = "UTC" ;
int Itime2(time) ;
    Itime2:units = "msec since 00:00:00" ;
    Itime2:time_zone = "UTC" ;
char Times(time, DateStrLen) ;
    Times:time_zone = "UTC" ;
float zeta(time, node) ;
    zeta:long_name = "Water Surface Elevation" ;
    zeta:units = "meters" ;
    zeta:positive = "up" ;
    zeta:standard_name = "sea_surface_height_above_geoid" ;
    zeta:grid = "SSH_Mesh" ;
    zeta:coordinates = "" ;
    zeta:type = "data" ;
float u(time, /usr/share/doc/HTML/index.htmlsiglay, nele) ;
    u:long_name = "Eastward Water Velocity" ;
    u:units = "meters s-1" ;
    u:grid = "fvcom_grid" ;
    u:type = "data" ;
float v(time, siglay, nele) ;
    v:long_name = "Northward Water Velocity" ;
    v:units = "meters s-1" ;
    v:grid = "fvcom_grid" ;
    v:type = "data" ;
float ua(time, nele) ;
    ua:long_name = "Vertically Averaged x-velocity" ;
    ua:units = "meters s-1" ;
    ua:grid = "fvcom_grid" ;
    ua:type = "data" ;
float va(time, nele) ;
    va:long_name = "Vertically Averaged y-velocity" ;
    va:units = "meters s-1" ;
    va:grid = "fvcom_grid" ;
    va:type = "data" ;
float w(time, siglev, nele) ;
    w:long_name = "Vertical Sigma Coordinate Velocity" ;
    w:units = "s-1" ;
    w:grid = "fvcom_grid" ;
    w:type = "data" ;
float km(time, siglev, node) ;
    km:long_name = "Turbulent Eddy Viscosity For Momentum" ;
    km:units = "m 2 s-1" ;
    km:grid = "fvcom_grid" ;
    km:coordinates = "" ;
```

```
        km:type = "data" ;
float kh(time, siglev, node) ;
        kh:long_name = "Turbulent Eddy Viscosity For Scalars" ;
        kh:units = "m 2 s-1" ;
        kh:grid = "fvcom_grid" ;
        kh:coordinates = "" ;
        kh:type = "data" ;
float kq(time, siglev, node) ;
        kq:long_name = "Turbulent Eddy Viscosity For Q2/Q2L" ;
        kq:units = "m 2 s-1" ;
        kq:grid = "fvcom_grid" ;
        kq:coordinates = "" ;
        kq:type = "data" ;
float q2(time, siglev, node) ;
        q2:long_name = "Turbulent Kinetic Energy" ;
        q2:units = "m2 s-2" ;
        q2:grid = "fvcom_grid" ;
        q2:coordinates = "" ;
        q2:type = "data" ;
float q2l(time, siglev, node) ;
        q2l:long_name = "Turbulent Kinetic Energy X Turbulent Macroscale" ;
        q2l:units = "m3 s-2" ;
        q2l:grid = "fvcom_grid" ;
        q2l:coordinates = "" ;
        q2l:type = "data" ;
float l(time, siglev, node) ;
        l:long_name = "Turbulent Macroscale" ;
        l:units = "m3 s-2" ;
        l:grid = "fvcom_grid" ;
        l:coordinates = "" ;
        l:type = "data" ;
float temp(time, siglay, node) ;
        temp:long_name = "temperature" ;
        temp:standard_name = "sea_water_temperature" ;
        temp:units = "degrees_C" ;
        temp:grid = "fvcom_grid" ;
        temp:coordinates = "" ;
        temp:type = "data" ;
float salinity(time, siglay, node) ;
        salinity:long_name = "salinity" ;
        salinity:standard_name = "sea_water_salinity" ;
        salinity:units = "1e-3" ;
        salinity:grid = "fvcom_grid" ;
        salinity:coordinates = "" ;
        salinity:type = "data" ;
float cor(nele) ;
        cor:long_name = "Coriolis Parameter" ;
        cor:units = "s-1" ;
        cor:grid = "fvcom_grid" ;
        cor:type = "data" ;
float cc_sponge(nele) ;
        cc_sponge:long_name = "Sponge Layer Parameter" ;
        cc_sponge:units = "nd" ;
        cc_sponge:grid = "fvcom_grid" ;
        cc_sponge:type = "data" ;
float et(time, node) ;
```

334

```
                et:long_name = "Water Surface Elevation At Last Timestep" ;
                et:units = "meters" ;
                et:positive = "up" ;
                et:standard_name = "sea_surface_elevation" ;
                et:grid = "SSH_Mesh" ;
                et:type = "data" ;
        float tmean1(siglay, node) ;
                tmean1:long_name = "mean initial temperature" ;
                tmean1:standard_name = "sea_water_temperature" ;
                tmean1:units = "degrees_C" ;
                tmean1:grid = "SigmaLayer_Mesh" ;
                tmean1:type = "data" ;
        float smean1(siglay, node) ;
                smean1:long_name = "mean initial salinity" ;
                smean1:standard_name = "sea_water_temperature" ;
                smean1:units = "1e-3" ;
                smean1:grid = "SigmaLayer_Mesh" ;
                smean1:type = "data" ;
        int obc_nodes(nobc) ;
                obc_nodes:long_name = "Open Boundary Node Number" ;
                obc_nodes:grid = "obc_grid" ;
        int obc_type(nobc) ;
                obc_type:long_name = "Open Boundary Type" ;
                obc_type:grid = "obc_grid" ;
        int lsf_nodes(nlsf) ;
                lsf_nodes:long_name = "Longshore Flow Node Number" ;
                lsf_nodes:grid = "lsf_grid" ;
        float wdf(nlsf) ;
                wdf:long_name = "Wind Driven Flow Adjustment Scaling" ;
                wdf:valid_range = "[0 1]" ;
                wdf:grid = "lsf_grid" ;
        float geo(nlsf) ;
                geo:long_name = "Thermal Wind Flow Adjustment Scaling" ;
                geo:valid_range = "[0 1]" ;
                geo:grid = "lsf_grid" ;

// global attributes:
                :title = "\'AN FVCOM CASE DESCRIPTION\' - note string must be in \'quotes\'" ;
                :institution = "School for Marine Science and Technology" ;
                :source = "FVCOM_3.0" ;
                :history = "model started at: 25/10/2010   11:36" ;
                :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
                :Conventions = "CF-1.0" ;
                :CoordinateSystem = "Cartesian" ;
                :CoordinateProjection = "none: A recognized reference coordinate for projtion for PROJ4" ;
}
```

5. Casename_tsobc.nc:

ncdump -h Casename_tsobc.nc

```
netcdf Casename_tsobc {
dimensions:
        nobc = 94 ;
        siglay = 40 ;
```

```
        siglev = 41 ;
        time = UNLIMITED ; // (14 currently)
        DateStrLen = 26 ;
variables:
        int obc_nodes(nobc) ;
                obc_nodes:long_name = "Open Boundary Node Number" ;
                obc_nodes:grid = "obc_grid" ;
        float obc_h(nobc) ;
                obc_h:long_name = "open boundary depth" ;
                obc_h:grid = "obc_grid" ;
        float obc_siglay(siglay, nobc) ;
                obc_siglay:long_name = "ocean_sigma/general_coordinate" ;
                obc_siglay:grid = "obc_grid" ;
        float obc_siglev(siglev, nobc) ;
                obc_siglev:long_name = "ocean_sigma/general_coordinate" ;
                obc_siglev:grid = "obc_grid" ;
        int iint(time) ;
                iint:long_name = "internal mode iteration number" ;
        float time(time) ;
                time:long_name = "time" ;
                time:units = "days since 1858-11-17 00:00:00" ;
                time:format = "modified julian day (MJD)" ;
                time:time_zone = "UTC" ;
        int Itime(time) ;
                Itime:units = "days since 1858-11-17 00:00:00" ;
                Itime:format = "modified julian day (MJD)" ;
                Itime:time_zone = "UTC" ;
        int Itime2(time) ;
                Itime2:units = "msec since 00:00:00" ;
                Itime2:time_zone = "UTC" ;
        char Times(time, DateStrLen) ;
                Times:time_zone = "UTC" ;
        float obc_temp(time, siglay, nobc) ;
                obc_temp:long_name = "sea_water_temperature" ;
                obc_temp:units = "Celcius" ;
                obc_temp:grid = "obc_grid" ;
        float obc_salinity(time, siglay, nobc) ;
                obc_salinity:long_name = "sea_water_salinity" ;
                obc_salinity:units = "PSU" ;
                obc_salinity:grid = "obc_grid" ;

// global attributes:
                :type = "FVCOM TIME SERIES OBC TS FILE" ;
                :title = "This data was transformed from an old FVCOM TS nudging file" ;
                :history = "FILE CREATED: 20100825T102843.977" ;
}
```

6. Casename_river.nc

ncdump -h Casename_river.nc

```
netcdf Casename_river {
dimensions:
        namelen = 80 ;
        rivers = 1 ;
```

```
        time = UNLIMITED ; // (11324 currently)
        DateStrLen = 26 ;
variables:
        char river_names(rivers, namelen) ;
        float time(time) ;
            time:long_name = "time" ;
            time:units = "days since 1858-11-17 00:00:00" ;
            time:format = "modified julian day (MJD)" ;
            time:time_zone = "UTC" ;
        int Itime(time) ;
            Itime:units = "days since 1858-11-17 00:00:00" ;
            Itime:format = "modified julian day (MJD)" ;
            Itime:time_zone = "UTC" ;
        int Itime2(time) ;
            Itime2:units = "msec since 00:00:00" ;
            Itime2:time_zone = "UTC" ;
        char Times(time, DateStrLen) ;
            Times:time_zone = "UTC" ;
        float river_flux(time, rivers) ;
            river_flux:long_name = "river runoff volume flux" ;
            river_flux:units = "m^3s^-1" ;
        float river_temp(time, rivers) ;
            river_temp:long_name = "river runoff temperature" ;
            river_temp:units = "Celsius" ;
        float river_salt(time, rivers) ;
            river_salt:long_name = "river runoff salinity" ;
            river_salt:units = "PSU" ;

// global attributes:
            :type = "FVCOM RIVER FORCING FILE" ;
            :title = "river" ;
            :website = "USGS" ;
            :history = "FVCOM-GoM3 model" ;
}
```

7. Casename_node_nest.nc

ncdump -h Casename_node_nest.nc

```
netcdf Casename_node_nest {
dimensions:
        time = UNLIMITED ; // (89281 currently)
        node = 70 ;
        siglev = 11 ;
        nele = 68 ;
        three = 3 ;
        siglay = 10 ;
variables:
        int Itime(time) ;
            Itime:units = "days since 1858-11-17 00:00:00" ;
            Itime:format = "modified julian day (MJD)" ;
            Itime:time_zone = "UTC" ;
        int Itime2(time) ;
            Itime2:units = "msec since 00:00:00" ;
            Itime2:time_zone = "UTC" ;
```

```
    float h(node) ;
        h:long_name = "Bathymetry" ;
        h:standard_name = "sea_floor_depth_below_geoid" ;
        h:units = "m" ;
        h:positive = "down" ;
        h:grid = "Bathymetry_Mesh" ;
        h:coordinates = "x y" ;
        h:type = "data" ;
    float hyw(time, siglev, node) ;
        hyw:long_name = "hydro static vertical velocity" ;
        hyw:units = "m/s" ;
        hyw:grid = "fvcom_grid" ;
        hyw:coordinates = "x y" ;
        hyw:type = "data" ;
    int iint(time) ;
        iint:long_name = "internal mode iteration number" ;
    float lat(node) ;
        lat:long_name = "nodal latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    float latc(nele) ;
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
    float lon(node) ;
        lon:long_name = "nodal longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lonc(nele) ;
        lonc:long_name = "zonal longitude" ;
        lonc:standard_name = "longitude" ;
        lonc:units = "degrees_east" ;
    int nprocs ;
        nprocs:long_name = "number of processors" ;
    int nv(three, nele) ;
        nv:long_name = "nodes surrounding element" ;
    int partition(nele) ;
        partition:long_name = "partition" ;
    float salinity(time, siglay, node) ;
        salinity:long_name = "salinity" ;
        salinity:standard_name = "sea_water_salinity" ;
        salinity:units = "1e-3" ;
        salinity:grid = "fvcom_grid" ;
        salinity:coordinates = "x y" ;
        salinity:type = "data" ;
    float siglay(siglay, node) ;
        siglay:long_name = "Sigma Layers" ;
        siglay:standard_name = "ocean_sigma/general_coordinate" ;
        siglay:positive = "up" ;
        siglay:valid_min = -1.f ;
        siglay:valid_max = 0.f ;
        siglay:formula_terms = "sigma: siglay eta: zeta depth: h" ;
    float siglev(siglev, node) ;
        siglev:long_name = "Sigma Levels" ;
        siglev:standard_name = "ocean_sigma/general_coordinate" ;
        siglev:positive = "up" ;
```

```
        siglev:valid_min = -1.f ;
        siglev:valid_max = 0.f ;
        siglev:formula_terms = "sigma:siglay eta: zeta depth: h" ;
float temp(time, siglay, node) ;
        temp:long_name = "temperature" ;
        temp:standard_name = "sea_water_temperature" ;
        temp:units = "degrees_C" ;
        temp:grid = "fvcom_grid" ;
        temp:coordinates = "x y" ;
        temp:type = "data" ;
float time(time) ;
        time:long_name = "time" ;
        time:units = "days since 1858-11-17 00:00:00" ;
        time:format = "modified julian day (MJD)" ;
        time:time_zone = "UTC" ;
float u(time, siglay, nele) ;
        u:long_name = "Eastward Water Velocity" ;
        u:units = "meters s-1" ;
        u:grid = "fvcom_grid" ;
        u:type = "data" ;
float ua(time, nele) ;
        ua:long_name = "Vertically Averaged x-velocity" ;
        ua:units = "meters s-1" ;
        ua:grid = "fvcom_grid" ;
        ua:type = "data" ;
float v(time, siglay, nele) ;
        v:long_name = "Northward Water Velocity" ;
        v:units = "meters s-1" ;
        v:grid = "fvcom_grid" ;
        v:type = "data" ;
float va(time, nele) ;
        va:long_name = "Vertically Averaged y-velocity" ;
        va:units = "meters s-1" ;
        va:grid = "fvcom_grid" ;
        va:type = "data" ;
float x(node) ;
        x:long_name = "nodal x-coordinate" ;
        x:units = "meters" ;
float xc(nele) ;
        xc:long_name = "zonal x-coordinate" ;
        xc:units = "meters" ;
float y(node) ;
        y:long_name = "nodal y-coordinate" ;
        y:units = "meters" ;
float yc(nele) ;
        yc:long_name = "zonal y-coordinate" ;
        yc:units = "meters" ;
float zeta(time, node) ;
        zeta:long_name = "Water Surface Elevation" ;
        zeta:units = "meters" ;
        zeta:positive = "up" ;
        zeta:standard_name = "sea_surface_elevation" ;
        zeta:grid = "SSH_Mesh" ;
        zeta:coordinates = "x y" ;
        zeta:type = "data" ;
```

// global attributes:
        :title = "FVCOM GOM3v6_nest NECOFS FORECAST UPDATE" ;
        :institution = "School for Marine Science and Technology" ;
        :source = "FVCOM_3.0" ;
        :history = "Wed Nov 3 09:43:49 2010: ncks -F -d siglay,1,10 -d siglev,1,11 noneed.nc
noneed1.nc\n",
        "Wed Nov 3 09:43:49 2010: ncks -F -d time,1 gom3v7_node_nest.nc noneed.nc\n",
        "model started at: 26/10/2010   16:08" ;
        :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
        :Conventions = "CF-1.0" ;
        :CoordinateSystem = "Cartesian" ;
        :CoordinateProjection = "init=nad83:1802" ;
        :Tidal_Forcing = "Tidal Forcing Time Series Title: JULIAN FVCOM TIDAL FORCING DATA
CREATED FROM OLD FILE TYPE: No comments found... this is mystery data!" ;
        :River_Forcing = "THERE ARE 49 RIVERS IN THIS MODEL.\n",
        "RIVER INFLOW IS ON THE nodes WHERE TEMPERATURE AND SALINITY ARE
calculated IN THE MODEL.\n",
        "THE FOLLOWING RIVER NAMES ARE USED:\n",
        "St_John_basin_ME\n",
        "Allagash_ME\n",
        "Aroostook_ME\n",
        "St_John_dickey_ME\n",
        "St_John_fish_ME\n",
        "St_Croix_ME\n",
        "Penobscot_ME\n",
        "Androscoggin_ME\n",
        "Saco_ME\n",
        "Kennebec_ME\n",
        "Merrimac_MA\n",
        "Spicket_MA\n",
        "Hudson_Fort_Edward_NY\n",
        "Hoosic_NY\n",
        "Mohawk_NY\n",
        "Lamprey_NH\n",
        "Winnicut_NH\n",
        "Exeter_NH\n",
        "Cocheco_NH\n",
        "Isinglass_NH\n",
        "Hunt_RI\n",
        "Pawtuxet_RI\n",
        "Pawcatuck_RI\n",
        "Moshassuck_RI\n",
        "Blackstone_RI\n",
        "Tenmile_RI\n",
        "Mill_RI\n",
        "Threemile_MA\n",
        "Wading_MA\n",
        "Taunton_MA\n",
        "Paskamanset_MA\n",
        "Quashnet_MA\n",
        "Jones_MA\n",
        "Indian_Head_MA\n",
        "Neponset_MA\n",
        "Charles_MA\n",
        "Aberjona_MA\n",
        "Alewife_MA\n",

```
                    "Saugus_MA\n",
                    "Ipswich_MA\n",
                    "Parker_MA\n",
                    "Connecticut_CT\n",
                    "Farmington_CT\n",
                    "Salmon_CT\n",
                    "Quinebaug_CT\n",
                    "Yantic_CT\n",
                    "Shetucket_CT\n",
                    "Housatonic_CT\n",
                    "Quinnipiac_CT" ;
            :GroundWater_Forcing = "GROUND WATER FORCING IS OFF!" ;
            :Surface_Heat_Forcing = "FVCOM variable surface heat forcing file:\n",
                    "FILE NAME:gom3v12_wnd.nc\n",
                    "SOURCE:FVCOM grid (unstructured) surface forcing\n",
                    "Unknown start date meta data format" ;
            :Surface_Wind_Forcing = "FVCOM variable surface Wind forcing:\n",
                    "FILE NAME:gom3v12_wnd.nc\n",
                    "SOURCE:FVCOM grid (unstructured) surface forcing\n",
                    "Unknown start date meta data format" ;
            :Surface_PrecipEvap_Forcing = "FVCOM periodic surface precip forcing:\n",
                    "FILE NAME:gom3v12_wnd.nc\n",
                    "SOURCE:FVCOM grid (unstructured) surface forcing\n",
                    "Unknown start date meta data format" ;
            :Special_Physical_processes = "long shore flow adjustment for thermal wind and wind driven
setup" ;
}
```

8. Casename_node_nest_wave.nc

ncdump -h Casename_node_nest_wave.nc

```
netcdf casename_node_nest_wave {
dimensions:
        nele = 68 ;
        node = 70 ;
        siglay = 40 ;
        siglev = 41 ;
        three = 3 ;
        time = UNLIMITED ; // (43201 currently)
        msc = 25 ;
        mdc = 24 ;
variables:
        int nprocs ;
                nprocs:long_name = "number of processors" ;
        int partition(nele) ;
                partition:long_name = "partition" ;
        float x(node) ;
                x:long_name = "nodal x-coordinate" ;
                x:units = "meters" ;
        float y(node) ;
                y:long_name = "nodal y-coordinate" ;
                y:units = "meters" ;
        float lon(node) ;
                lon:long_name = "nodal longitude" ;
```

```
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
float lat(node) ;
        lat:long_name = "nodal latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
float xc(nele) ;
        xc:long_name = "zonal x-coordinate" ;
        xc:units = "meters" ;
float yc(nele) ;
        yc:long_name = "zonal y-coordinate" ;
        yc:units = "meters" ;
float lonc(nele) ;
        lonc:long_name = "zonal longitude" ;
        lonc:standard_name = "longitude" ;
        lonc:units = "degrees_east" ;
float latc(nele) ;
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
float siglay(siglay, node) ;
        siglay:long_name = "Sigma Layers" ;
        siglay:standard_name = "ocean_sigma/general_coordinate" ;
        siglay:positive = "up" ;
        siglay:valid_min = -1.f ;
        siglay:valid_max = 0.f ;
        siglay:formula_terms = "sigma: siglay eta: zeta depth: h" ;
float siglev(siglev, node) ;
        siglev:long_name = "Sigma Levels" ;
        siglev:standard_name = "ocean_sigma/general_coordinate" ;
        siglev:positive = "up" ;
        siglev:valid_min = -1.f ;
        siglev:valid_max = 0.f ;
        siglev:formula_terms = "sigma:siglay eta: zeta depth: h" ;
float h(node) ;
        h:long_name = "Bathymetry" ;
        h:standard_name = "sea_floor_depth_below_geoid" ;
        h:units = "m" ;
        h:positive = "down" ;
        h:grid = "Bathymetry_Mesh" ;
        h:coordinates = "x y" ;
        h:type = "data" ;
int nv(three, nele) ;
        nv:long_name = "nodes surrounding element" ;
int iint(time) ;
        iint:long_name = "internal mode iteration number" ;
float time(time) ;
        time:long_name = "time" ;
        time:units = "days since 1858-11-17 00:00:00" ;
        time:format = "modified julian day (MJD)" ;
        time:time_zone = "UTC" ;
int Itime(time) ;
        Itime:units = "days since 1858-11-17 00:00:00" ;
        Itime:format = "modified julian day (MJD)" ;
        Itime:time_zone = "UTC" ;
int Itime2(time) ;
```

```
        Itime2:units = "msec since 00:00:00" ;
        Itime2:time_zone = "UTC" ;
    float ac2(time, mdc, msc, node) ;
        ac2:long_name = "Wave Spectral Density" ;
        ac2:units = "DONTKNOW" ;
        ac2:grid = "fvcom_grid" ;
        ac2:type = "data" ;

// global attributes:
        :title = "FVCOM GOM3v6_nest NECOFS FORECAST UPDATE" ;
        :institution = "School for Marine Science and Technology" ;
        :source = "FVCOM_3.0" ;
        :history = "model started at: 16/08/2011   10:07" ;
        :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
        :Conventions = "CF-1.0" ;
        :CoordinateSystem = "Cartesian" ;
        :CoordinateProjection = "init=nad83:1802" ;
        :Tidal_Forcing = "TIDAL ELEVATION FORCING IS OFF!" ;
        :River_Forcing = "THERE ARE NO RIVERS IN THIS MODEL" ;
        :GroundWater_Forcing = "GROUND WATER FORCING IS OFF!" ;
        :Surface_Heat_Forcing = "SURFACE HEAT FORCING IS OFF" ;
        :Surface_Wind_Forcing = "FVCOM variable surface Wind forcing:\n",
            "FILE NAME:wrf_for.nc\n",
            "SOURCE:wrf2fvcom version 0.13 (2007-07-19) (Bulk method: COARE 2.6Z)\n",
            "MET DATA START DATE:2007-01-01_00:00:00" ;
        :Surface_PrecipEvap_Forcing = "SURFACE PRECIPITATION FORCING IS OFF" ;
}
```

9. Casename_sst.nc

ncdump -h Casename_sst.nc

```
netcdf Casename_sst {
dimensions:
    nele = 91258 ;
    node = 48860 ;
    three = 3 ;
    time = UNLIMITED ; // (367 currently)
    DateStrLen = 26 ;
variables:
    int nprocs ;
        nprocs:long_name = "number of processors" ;
    int partition(nele) ;
        partition:long_name = "partition" ;
    float x(node) ;
        x:long_name = "nodal x-coordinate" ;
        x:units = "meters" ;
    float y(node) ;
        y:long_name = "nodal y-coordinate" ;
        y:units = "meters" ;
    float lon(node) ;
        lon:long_name = "nodal longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lat(node) ;
```

```
        lat:long_name = "nodal latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    float xc(nele) ;
        xc:long_name = "zonal x-coordinate" ;
        xc:units = "meters" ;
    float yc(nele) ;
        yc:long_name = "zonal y-coordinate" ;
        yc:units = "meters" ;
    float lonc(nele) ;
        lonc:long_name = "zonal longitude" ;
        lonc:standard_name = "longitude" ;
        lonc:units = "degrees_east" ;
    float latc(nele) ;
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
    int nv(three, nele) ;
        nv:long_name = "nodes surrounding element" ;
    int iint(time) ;
        iint:long_name = "internal mode iteration number" ;
    float time(time) ;
        time:long_name = "time" ;
        time:units = "days since 1858-11-17 00:00:00" ;
        time:format = "modified julian day (MJD)" ;
        time:time_zone = "UTC" ;
    int Itime(time) ;
        Itime:units = "days since 1858-11-17 00:00:00" ;
        Itime:format = "modified julian day (MJD)" ;
        Itime:time_zone = "UTC" ;
    int Itime2(time) ;
        Itime2:units = "msec since 00:00:00" ;
        Itime2:time_zone = "UTC" ;
    char Times(time, DateStrLen) ;
        Times:time_zone = "UTC" ;
    float sst(time, node) ;
        sst:long_name = "Sea Surface Temperature" ;
        sst:units = "celcius" ;
        sst:grid = "fvcom_grid" ;
        sst:type = "data" ;

// global attributes:
        :title = "\'AN FVCOM CASE DESCRIPTION\' - note string must be in \'quotes\'" ;
        :institution = "School for Marine Science and Technology" ;
        :source = "FVCOM_3.0" ;
        :history = "Mon Oct 25 11:49:56 2010: ncrcat -d time,0,366 sst.nc sst2005.nc\n",
            "model started at: 25/10/2010   11:40" ;
        :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
        :Conventions = "CF-1.0" ;
        :CoordinateSystem = "Cartesian" ;
        :CoordinateProjection = "init=nad83:1802" ;
        :nco_openmp_thread_number = 1 ;
}
```

10. Casename_hvc.nc

ncdump -h Casename_hvc.nc

```
netcdf Casename_hvc {
dimensions:
    node = 48860 ;
    nele = 91258 ;
    three = 3 ;
variables:
    float x(node) ;
        x:long_name = "nodal x-coordinate" ;
        x:units = "meters" ;
    float y(node) ;
        y:long_name = "nodal y-coordinate" ;
        y:units = "meters" ;
    float lon(node) ;
        lon:long_name = "nodal longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lat(node) ;
        lat:long_name = "nodal latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    int nv(three, nele) ;
        nv:long_name = "nodes surrounding element" ;
    float nn_hvc(node) ;
        nn_hvc:long_name = "NN_HVC" ;
        nn_hvc:units = "m+2 s-1" ;
        nn_hvc:grid = "fvcom_grid" ;
        nn_hvc:type = "data" ;
    float cc_hvc(nele) ;
        cc_hvc:long_name = "CC_HVC" ;
        cc_hvc:units = "m+2 s-1" ;
        cc_hvc:grid = "fvcom_grid" ;
        cc_hvc:type = "data" ;

// global attributes:
        :institution = "School for Marine Science and Technology" ;
        :source = "FVCOM_2.6" ;
        :history = "model started at: 29/09/2010   19:18" ;
        :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
        :Conventions = "CF-1.0" ;
        :CoordinateProjection = "none: A recognized reference coordinate for projtion for PROJ4" ;
}
```

11. Casename_brf.nc

ncdump -h Casename_brf.nc

```
netcdf Casename_brf {
dimensions:
    node = 48860 ;
    nele = 91258 ;
```

```
        three = 3 ;
variables:
        float x(node) ;
                x:long_name = "nodal x-coordinate" ;
                x:units = "meters" ;
        float y(node) ;
                y:long_name = "nodal y-coordinate" ;
                y:units = "meters" ;
        float lon(node) ;
                lon:long_name = "nodal longitude" ;
                lon:standard_name = "longitude" ;
                lon:units = "degrees_east" ;
        float lat(node) ;
                lat:long_name = "nodal latitude" ;
                lat:standard_name = "latitude" ;
                lat:units = "degrees_north" ;
        int nv(three, nele) ;
                nv:long_name = "nodes surrounding element" ;
        float z0b(nele) ;
                z0b:long_name = "Bottom Roughness Lengthscale" ;
                z0b:units = "m" ;
                z0b:grid = "fvcom_grid" ;
                z0b:type = "data" ;

// global attributes:
                :institution = "School for Marine Science and Technology" ;
                :source = "FVCOM_2.6" ;
                :history = "model started at: 29/09/2010   19:18" ;
                :references = "http://fvcom.smast.umassd.edu, http://codfish.smast.umassd.edu" ;
                :Conventions = "CF-1.0" ;
                :CoordinateProjection = "none: A recognized reference coordinate for projtion for PROJ4" ;
}
```

12. Casename_lag_start.nc

ncdump -h Casename_lag_start.nc

```
netcdf Casename_lag_start {
dimensions:
        nparticles = 10 ;
variables:
        float x(nparticles) ;
                x:long_name = "particle x position" ;
                x:units = "m" ;
        float y(nparticles) ;
                y:long_name = "particle y position" ;
                y:units = "m" ;
        float z(nparticles) ;
                z:long_name = "particle z position" ;
                z:units = "m" ;
        float tbeg(nparticles) ;
                tbeg:long_name = "particle release time" ;
                tbeg:units = "days since 0.0" ;
                tbeg:time_zone = "none" ;
        float tend(nparticles) ;
```

```
            tend:long_name = "particle freeze time" ;
            tend:units = "days since 0.0" ;
            tend:time_zone = "none" ;
        float pathlength(nparticles) ;
            pathlength:long_name = "particle integrated path length" ;
            pathlength:units = "m" ;
        int group(nparticles) ;
            group:long_name = "particle group" ;
            group:units = "-" ;
        int mark(nparticles) ;
            mark:long_name = "particle marker (0=in domain)" ;
            mark:units = "-" ;

// global attributes:
            :info_string = "test tracking" ;
            :dump_counter = 0 ;
            :t_last_dump = 0.f ;
            :number_particles = 10 ;
}
```

13. Casename_hfx.nc

ncdump -h Casename_hfx.nc

```
netcdf Casename_hfx {
dimensions:
        node = 90267 ;
        nele = 174474 ;
        three = 3 ;
        time = UNLIMITED ; // (17 currently)
        DateStrLen = 26 ;
variables:
        int Itime(time) ;
            Itime:units = "days since 1858-11-17 00:00:00" ;
            Itime:format = "modified julian day(MJD)" ;
            Itime:time_zone = "UTC" ;
        int Itime2(time) ;
            Itime2:units = "msec since 00:00:00" ;
            Itime2:time_zone = "UTC" ;
            Itime2:long_name = "time" ;
        float time(time) ;
            time:units = "days since 1858-11-17 00:00:00" ;
            time:format = "modified julian day(MJD)" ;
            time:time_zone = "UTC" ;
        char Times(time, DateStrLen) ;
            Times:long_name = "Calendar Date" ;
            Times:format = "String: Calendar Time" ;
            Times:time_zone = "UTC" ;
        int nv(three, nele) ;
            nv:long_name = "nodes surrounding elements" ;
        float lat(node) ;
            lat:long_name = "Nodal latitude" ;
            lat:standard_name = "latitude" ;
            lat:units = "degrees_north" ;
        float lon(node) ;
```

```
        lon:long_name = "Nodal longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
float latc(nele) ;
        latc:long_name = "zonal latitude" ;
        latc:standard_name = "latitude" ;
        latc:units = "degrees_north" ;
float lonc(nele) ;
        lonc:long_name = "zonal longitude" ;
        lonc:standard_name = "longitude" ;
        lonc:units = "degrees_east" ;
float air_temperature(time, node) ;
        air_temperature:long_name = "Surface air temperature" ;
        air_temperature:units = "Celsius Degree" ;
        air_temperature:grid = "fvcom_grid" ;
        air_temperature:coordinates = "FVCOM Spheric coordinates" ;
        air_temperature:type = "data" ;
float relative_humidity(time, node) ;
        relative_humidity:long_name = "surface air relative humidity" ;
        relative_humidity:units = "percentage" ;
        relative_humidity:grid = "fvcom_grid" ;
        relative_humidity:coordinates = "FVCOM Spheric coordinates" ;
        relative_humidity:type = "data" ;
float long_wave(time, node) ;
        long_wave:long_name = "Downward solar longtwave radiation flux" ;
        long_wave:units = "Watts meter-2" ;
        long_wave:grid = "fvcom_grid" ;
        long_wave:coordinates = "FVCOM Spheric coordinates" ;
        long_wave:type = "data" ;
        long_wave:positive = "downward flux, heating" ;
        long_wave:negative = "upward flux, cooling" ;
float short_wave(time, node) ;
        short_wave:long_name = "Net solar shortwave radiation flux" ;
        short_wave:units = "Watts meter-2" ;
        short_wave:grid = "fvcom_grid" ;
        short_wave:coordinates = "FVCOM Spheric coordinates" ;
        short_wave:type = "data" ;
        short_wave:positive = "downward flux, heating" ;
        short_wave:negative = "upward flux, cooling" ;
float net_heat_flux(time, node) ;
        net_heat_flux:long_name = "Net surface heat flux" ;
        net_heat_flux:units = "Watts meter-2" ;
        net_heat_flux:grid = "fvcom_grid" ;
        net_heat_flux:coordinates = "FVCOM Spheric coordinates" ;
        net_heat_flux:type = "data" ;
        net_heat_flux:positive = "downward flux, heating" ;
        net_heat_flux:negative = "upward flux, cooling" ;
float air_pressure(time, node) ;
        air_pressure:long_name = "surface air pressure" ;
        air_pressure:units = "Pa" ;
        air_pressure:grid = "fvcom_grid" ;
        air_pressure:coordinates = "FVCOM Spheric coordinates" ;
        air_pressure:type = "data" ;

// global attributes:
        :type = "operational products" ;
```

```
            :title = "surface forcing netCDF file" ;
            :source = "FVCOM grid (unstructured) surface forcing" ;
            :grib2_file = "Grib2 file: " ;
            :grid_info = "On FVCOM grid, using remesh spatial interpolation" ;
            :model_grid_file = "Ocean Model grid file: " ;
            :history = "Created at time 00:44  10/18/2011" ;
            :reference = "Created by NOAA" ;
}
```

Chapter 19: FVCOM Test Cases

To help users learn how to use FVCOM, we have included several simple examples here. Case 1: Tidally driven flooding/drying process in a semi-enclosed channel. Case 2: Freshwater discharge on an idealized continental shelf. Case 3: Wave-current-sediment interaction in an idealized inlet. Case 4: A 1-D ice simulation experiment. Case 5: Lock exchanges in a 3-D box domain. Case 6: an EnKF filter experiment for tidal oscillation problems in a circular domain. (At this time, only the first two test cases are included here. The remaining test cases will be added to the manual as they are added to the core FVCOM code.)

Case 1: Tidally-driven flooding/drying process in a semi-enclosed channel

This case was selected when we tested the wet/dry treatment in an estuary. The results were written up as a manuscript entitled "A 3-Dimensional, Unstructured Grid, Finite-Volume Wet/Dry Point Treatment Method for FVCOM". The unpublished manuscript is available. For users who are interested in knowing the details, please contact Chen at c1chen@umassd.edu. The water is homogeneous in this case, with no freshwater riverine or groundwater input and no atmospheric surface.

a) Design of the numerical experiment

Numerical experiments were conducted for an idealized semi-enclosed channel with a width of 3 km at the bottom, a length of 30 km, a constant depth of 10 m and a lateral slope of about 0.033 (Figure 19.1). This channel is oriented east to west, with connection to a relatively wide and flat-bottom shelf to the east and inter-tidal zones on the northern and southern channel edges. The inter-tidal zone is distributed symmetrically to the channel axis with a constant slope of α and a width of 2 km.

The computational domain was configured with unstructured triangular grids in the horizontal and σ-levels in the vertical. Numerical experiments were conducted for cases with different horizontal and vertical resolutions. The comparison between these cases was made based on differences from a standard run with a horizontal resolution of about

500 m in the channel, 600 to 1000 m on the shelf, and 600 to 900 m over the inter-tidal zone (Figure 19.1).

The model was forced by an M_2 tidal oscillation with amplitude ζ_o at the open boundary of the outer shelf. This oscillation creates a surface gravity wave that propagates into the channel and reflects back after it reaches the solid wall at the upstream end. For a given tidal elevation at this open boundary, the velocity in triangular elements connected to the boundary and water transport flowing out of the computational domain is determined through the incompressible continuity equation.
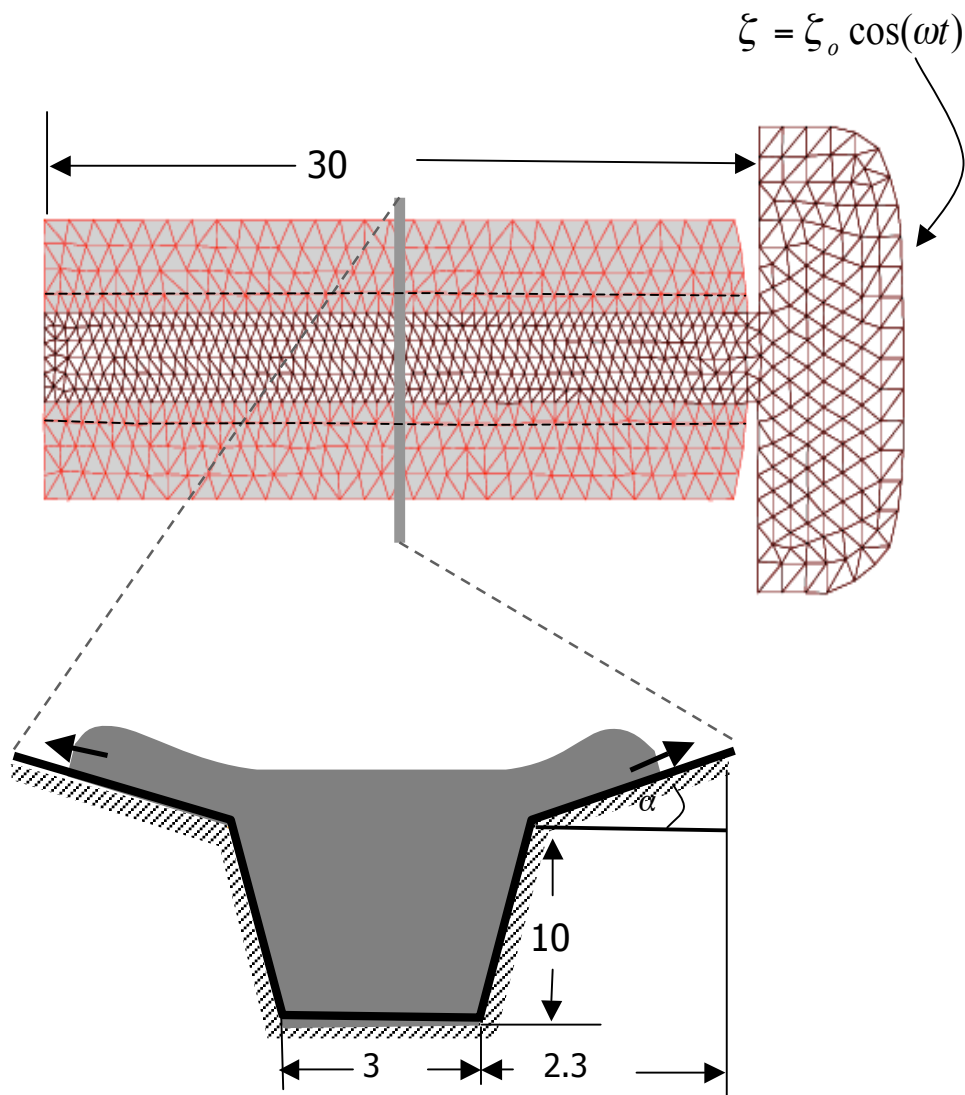
Figure 19.1: Unstructured triangular grid for the standard run plus a view of the cross-channel section. Dashed line along the channel indicates the edge of the channel connected to the inter-tidal zone. Horizontal distances in *km* and vertical depth in *m*.

b) The input files

    All files required to run this case are in the directory named "examples/estuary". In the subdirectory named "run", there is a namelist file "*tst.run.nml*". In the subdirectory named "tstinp", there are nine input files. Two are in NetCDF format and six are in ASCII format. One is the namelist for the river input. Following the procedures described in Chapter 17, it should be easy to set up and run this case.

Case 2: Freshwater discharge over an idealized continental shelf

a) Design of the numerical experiment

    Consider the linear sloping continental shelf shown in Figure 19.2. The freshwater is discharged onto the shelf from a point source at a distance of 200 km from the origin. The freshwater discharge rate Q is specified as 1000 m^3/s and the background salinity $S =$ 30 PSU. The atmospheric surface forcing and boundary tidal forcing are both set to zero.
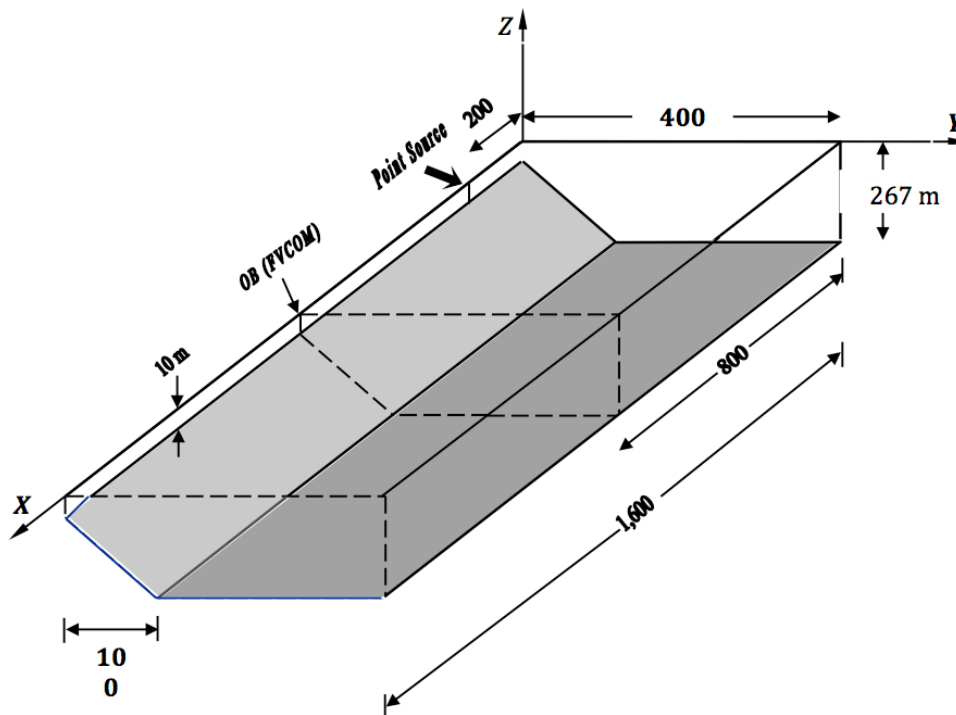
Fig. 19.2: An idealized linear slope continental shelf with a freshwater discharge at a point source. Horizontal distance is in *km* and vertical depth is in *m*.

The numerical domain is configured with unstructured triangular grids with a resolution of 20 km (Fig. 19.3). Ten sigma levels are used in the vertical. The open boundary is located at 800 km downstream from the origin, at which a gravity wave radiation boundary condition is specified to allow the wave energy to propagate out of the computational domain with minimum reflection. The numerical grid is shown below:
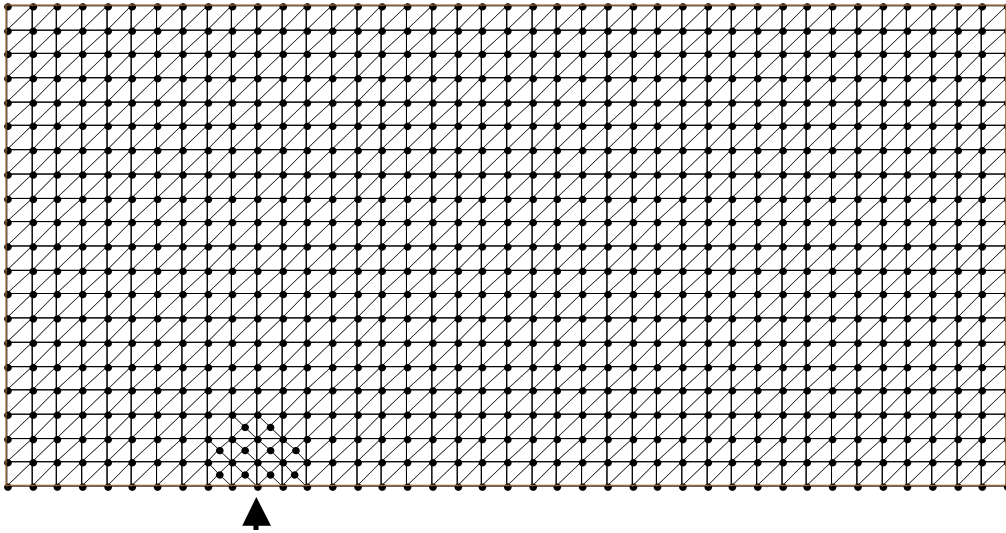
Fig. 19.3: Unstructured triangular grid for the case 2 experiment. The arrow indicates the location of the freshwater discharge.

b) The input files

All files required to run this case are in the directory named: "examples/river_plume". In the subdirectory named "run", there is a namelist file "*chn.run.nml*". In the subdirectory named "tstinp", there are seven input files. One is in NetCDF format and six are in ASCII format. Following the procedures described in Chapter 17, it should be easy to set up and run this case.

Case 3: Wave-current-sediment interaction in an inlet

a) Design of the numerical experiment

Consider an idealized tidally-driven coastal inlet. The geometrical domain of this inlet features a semi-enclosed rectangular basin with a width of 15 km and a length of 14 km. The initial water depth is specified to be 15 m at the open boundary ($y = 0$) and linearly

decreases to 4 *m* at the northern end (y = 15 *km*). The initial bed thickness is specified to be 10 m everywhere. A wall is placed across the middle line of the basin, with a 2-*km* wide water exit at the center.
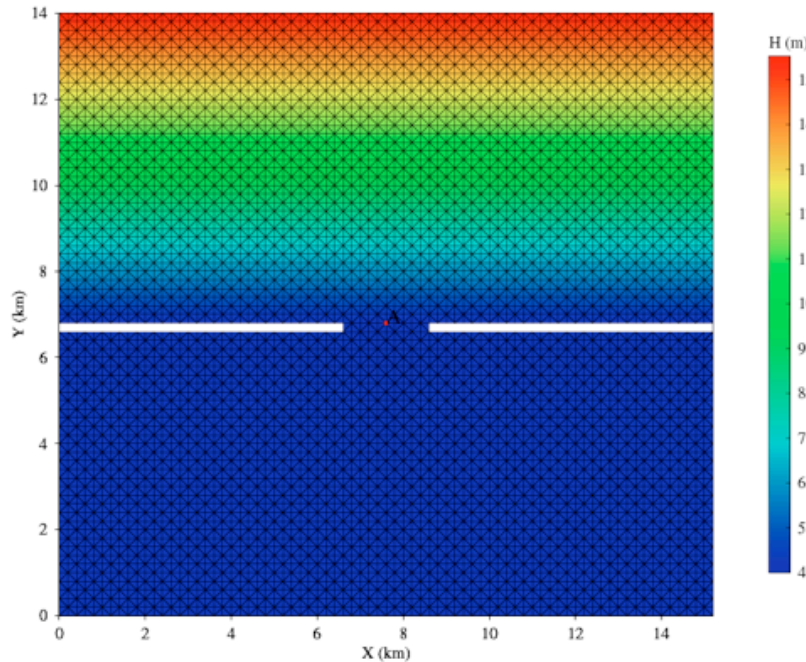
Figure 19.4: The triangular grid used for the inlet test problem.

The basin is configured with non-overlapped triangular grid with total numbers of 5467 for nodes and 10508 for cells. A 1-*m* amplitude M_2 tide is specified at the open boundary (y = 0 km). Surface waves with 1-*m* height and 10-s period are also imposed at the open boundary and propagate southward towards the coast.

b) The input files

All files required to run this case are in the directory named: "Examples/Inlet". In the subdirectory named "run", there is a namelist file "inlet.run.nml". In the subdirectory named "input", there are nine input files. One is in NetCDF format and eight are in ASCII format.

Case 4: A 1-D ice simulation experiment

a) Design of the numerical experiment

Consider a thermal-driven one-dimensional ice formation problem. The input forcing variables are: air temperature (°C), specific humidity (kg/kg), net short wave radiation (w/m^2) and cloud cover. The vertical profiles of water temperature and salinity at the initial were specified using the observation at 147.325°W and 76.82124°N, which was obtained from the PHC3.0 database. In this case, no wind and sea level air pressure were considered, and they could be easily added if needed. The initial condition for the ice model was specified using the default setup of CICE.

b) The input files

All files required to run this case are in the directory named: "Examples/Sea ice". In this directory, there is a namelist file "*N1D.run.nml*". In the subdirectory named "input_1D_new", there are twelve input files. Three are in NetCDF format and nine are in ASCII format. Following the procedures described in Chapter 17, it should be easy to set up and run this case.

Case 5: Lock-exchanges in a rectangular channel

a) Design of the numerical experiment

Consider a lock exchange problem in a rectangular tank filled with two fluids of different densities (hereafter referred as light and heavy) that are separated initially by a vertical gate at the tank center (Lai et al., 2010a). The tank has a length (*L*) of 0.8 m, a width (*W*) of 0.008 m and a static water depth (*H*) of 0.1 m. Assume that the density (ρ) is linearly proportional to salinity (*S*) given by

$$\rho = 999.972 \times (1 + 0.75 \times 10^{-3} S),$$

and $\rho_1 = 999.972$ kg/m^3 and $\rho_2 = 1000.991371$ kg/m^3. This gives a reduced gravity (g') as

$$g' = g\Delta\rho / \rho_o = 0.01 \text{ m/s}^2$$

where ρ_o is the reference density specified as 1000 kg/m^3 and g is the gravitational acceleration constant with a value of 9.81 m/s^2.

The rectangular tank was configured with non-overlapped triangular meshes in the horizontal and uniform layers in the vertical. In the horizontal, the total number of triangles is 400 (x-axis) ×8 (y-axis). In the vertical (z-axis), it was divided into 100 layers. This configuration provides a resolution of 0.2 cm in the horizontal and 0.1 cm in the vertical. The internal time step used in this experiment is 0.003 sec for the inviscid case and 0.001 sec for the viscous case, with an internal-external mode split ratio of 10. Two-dimension flow (no variation in y) was implemented using the methods outlined above. The model was integrated for three minutes for both inviscid and viscous (constant background molecular viscosity) conditions.

b) The input files

All files required to run this case are in the directory named: "Examples/Lock-Exchange". In this directory named "run", there is a namelist file "*tst.run.nml*". In the subdirectory named "INPDIR", there are eight input files. Two are in NetCDF format and six are in ASCII format.

Case 6: An EnKF filter experiment for tidal oscillation problems in a circular domain.

a) Design of the numerical experiment

Consider a tidal oscillation problem in an idealized flat bottom circular shelf. The water depth is 10 m everywhere and the tidal forcing at the open boundary is given by a M$_2$ tidal elevation with amplitude of 1 m. In this experiment, the model was first spin up under initial conditions of zero velocity and sea level. When the model reaches an equilibrium state, the model output was treated as "true state". Then, the model was re-run with wrong initial condition with Ensemble Kalman Filter (EnKF) assimilation. The 20 ensembles are created from the backward model output starting on day 10. The observations were constructed by extracting the value at selected sites from the "true state". Twin experiments with EnKF were made to test the effectiveness of EnKF in data assimilation for coastal ocean problems.

b) The input files

All files required to run this case are in the directory named: "Examples/KFs". In this directory, there is a namelist file "*readme.txt*". In the subdirectory named "true_run/input/", there are eight input files. One is in NetCDF format and seven are in ASCII format
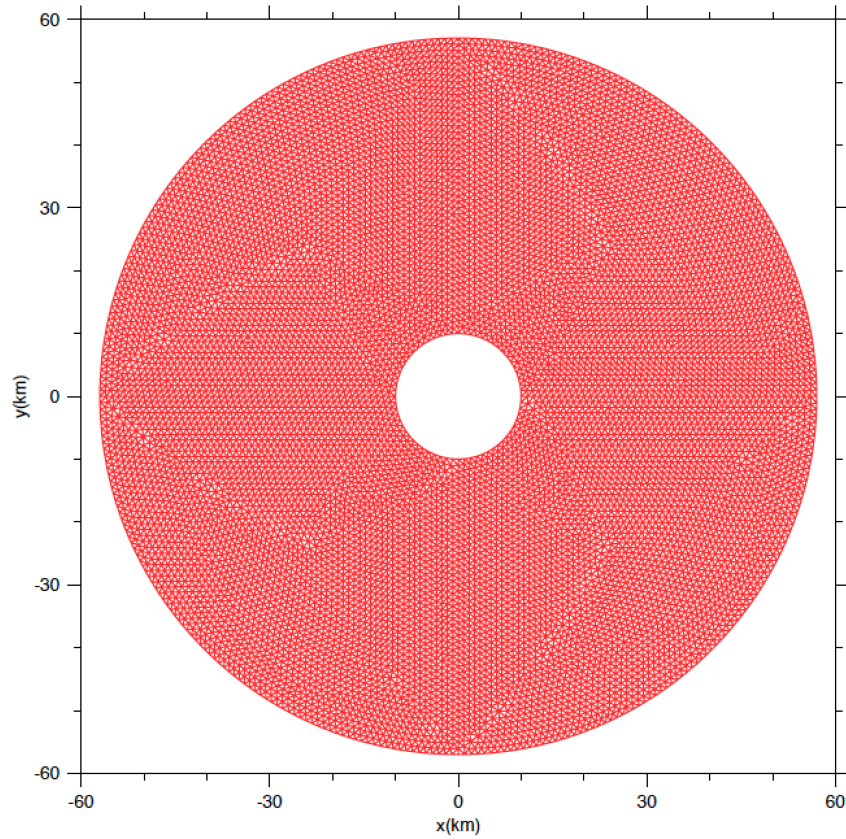
Fig. 19.5: The triangular grid used for the EnKF's experiment.

Chapter 20: Unstructured Triangular Mesh Generation

FVCOM uses unstructured triangular grids and since no automatic mesh generator is supplied with the coding, users must use alternate software to build the mesh. There are a multitude of mesh generators available. Some are open source and several are commercial. Any software which is capable of generating two-dimensional, unstructured triangular grids would work for FVCOM. However, high powered meshing software designed for complex 3-D grids, such as Gridgen, ICEM, or GAMBIT are expensive, difficult to learn, and are unnecessarily powerful. We have listed some recommended mesh generators on the FVCOM users website.

At MEDM/SMAST, we primarily use the SMS (Surface Water Model System) to generate unstructured grids in FVCOM. SMS is module-based commercial software that can be purchased in whole or by module. Only three modules are necessary for mesh generation. They are the Mesh Module, the Map Module, and the Scatter Module. The beta version can be downloaded from the SMS website:

   http://www.ems-i.com/SMS/SMS_Overview/sms_overview.html/

In this document, we will guide you through an example mesh generation using SMS to teach the basic procedure of generating meshes for FVCOM. The basic process is similar for either Cartesian or spherical coordinate system. The example given here is for generating a grid in Cartesian coordinate since this is normally used in coastal ocean applications.

20.1 Data Preparation

To generate an unstructured grid, users need to have first acquired relevant data including: 1) coastlines and 2) bathymetry.

a) Coastline preparation

FVCOM can be run in either spherical or Cartesian coordinates. For applications in the coastal ocean, we recommend that users use Cartesian coordinates. In general, the coastline data are in a geographic format of longitude and latitude. A projection program

is needed to convert the longitude and latitude to the Cartesian coordinates relative to a selected reference point. Procedures to prepare the coastline data are given below.

Step 1: Download the coastline data

Recommended website: http://www.ngdc.noaa.gov/mgg/geodas/geodas.html.

NOAA has a GEODAS CD available with GEODAS coastline extractor.

The format of the download coastline data is as follows:

```
1   41.5756   -70.5038
2   41.5756   -70.5041
2   41.5757   -70.5043
2   41.5760   -70.5043
2   41.5761   -70.5042
2   41.5761   -70.5039
2   41.5760   -70.5036
2   41.5757   -70.5036
3   41.5756   -70.5038
```

Column 1: the data type: 1-start point; 2-points between start and end points; 3-end point.

Column 2: Latitude

Column 3: Longitude

Step 2: Convert the longitude and latitude to Cartesian x-y coordinates

- Download the Cartographic Projection Tool called "**Proj**" from website http://proj.maptools.org/ and install it.

- Use "Proj" to convert the longitude and latitude format data to the *x-y* coordinate.

The format of the output data file looks like

```
871884.3374      -139646.0237
871859.3184      -139645.9259
871842.6826      -139634.7546
871842.8129      -139601.4361
871851.1959      -139590.3625
871876.2146      -139590.4603
871901.19        -139601.6641
871901.06        -139634.9826
871884.3374      -139646.0237
```

Column 1: x Column 2: y

Note: when "Proj" is used, be sure to keep the x and y locations of the reference point used for the projection.   These will be required if you wish to convert the x and y locations back to Lat/Lon.

3. Insert the 1st column (data type) from the original coastline data (latitude and longitude format) into the converted x-y data file.

"Proj" can only run with an input file of longitude and latitude data, so after the projection has been completed, we need to add the 1st column of the original data into the projected data to keep the identification of the data type.

4. Run "ReadCST.f" to create the coastline file format compatible with SMS.
We have a simple Fortran 77 program called "readcst_new.f" on the FVCOM user website. Users can use it or can write a very simple Matlab program to do it. The format of the resulting output file is:

```
COAST
   11  0.0
    9    0
         871884.3125         -139646.0313   0.0
         871859.3125         -139645.9219   0.0
         871842.6875         -139634.7500   0.0
         871842.8125         -139601.4375   0.0
         871851.1875         -139590.3594   0.0
         871876.1875         -139590.4531   0.0
         871901.1875         -139601.6719   0.0
         871901.0625         -139634.9844   0.0
         871884.3125         -139646.0313   0.0
   11    0
         871775.5625         -139734.4531   0.0
         871742.2500         -139723.2188   0.0
         871720.3125         -139712.0000   0.0
         871700.6875         -139689.7344   0.0
         871700.7500         -139667.5156   0.0
         871784.1875         -139667.8438   0.0
         871825.8125         -139679.1094   0.0
         871842.4375         -139701.3906   0.0
         871834.0000         -139723.5781   0.0
         871820.2500         -139734.6094   0.0
         871775.5625         -139734.4531   0.0
```

b) Bathymetric data preparation

The bathymetric data can be downloaded from the USGS bathymetric database or obtained from other data sources. In general, the data format is:

Latitude, Longitude, Depth

The same procedure described in the coastline preparation needs to be used here to convert the bathymetric data into the Cartesian x and y coordinates. Final data for SMS should look like:

```
870118.000 -139242.750 1.770
870812.875 -139245.594 0.500
870116.063 -139705.547 1.880
870811.000 -139708.391 0.500
871853.500 -139712.547 0.710
868029.063 -140159.375 0.160
870114.125 -140168.328 1.900
870809.125 -140171.188 0.630
871504.188 -140173.969 0.880
872199.188 -140176.688 0.890
868027.000 -140622.063 0.030
870112.250 -140631.016 1.910
```

Column 1: *x* (meters); Column 2: *y* (meters); Column 3: depth (meters).

Note: FVCOM requires that the water depth is positive for wet points.

Now, you are ready to run "SMS".

20.2 Grid Generation

1. Click "SMS", select "Start Using SMS"
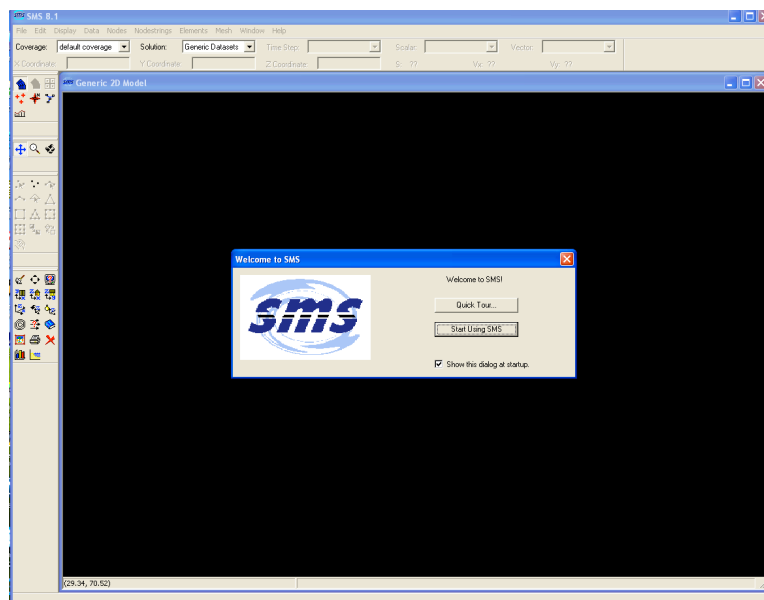
2. Go to "File" at the upper left corner. Select "Open" and then find the *.cst" file in your computer and open it. Select "ADCIRC" and click on it. Your coastline data should appear on the screen. This is an example of Waquoit Bay on the southern coast of Cape Cod, Massachusetts.
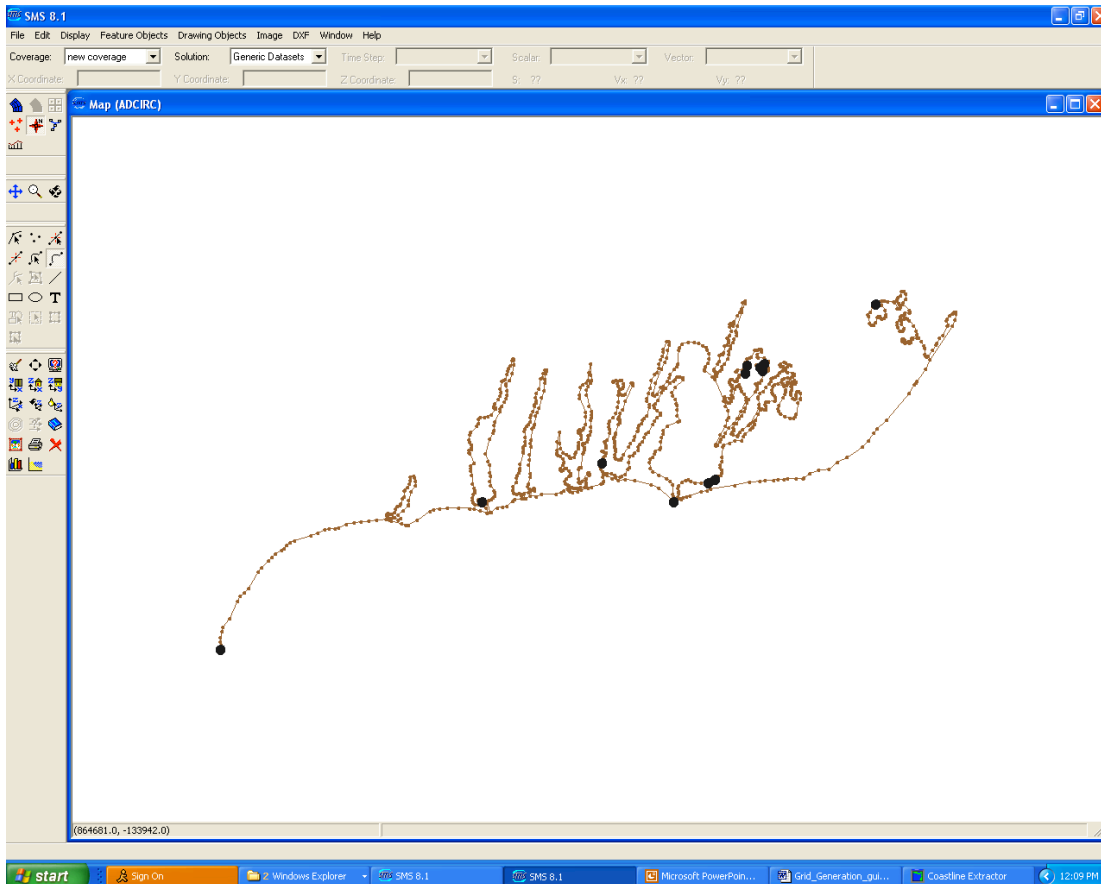
3. Create an initial open boundary

Specify the open boundary line user "Create Feature Arc".
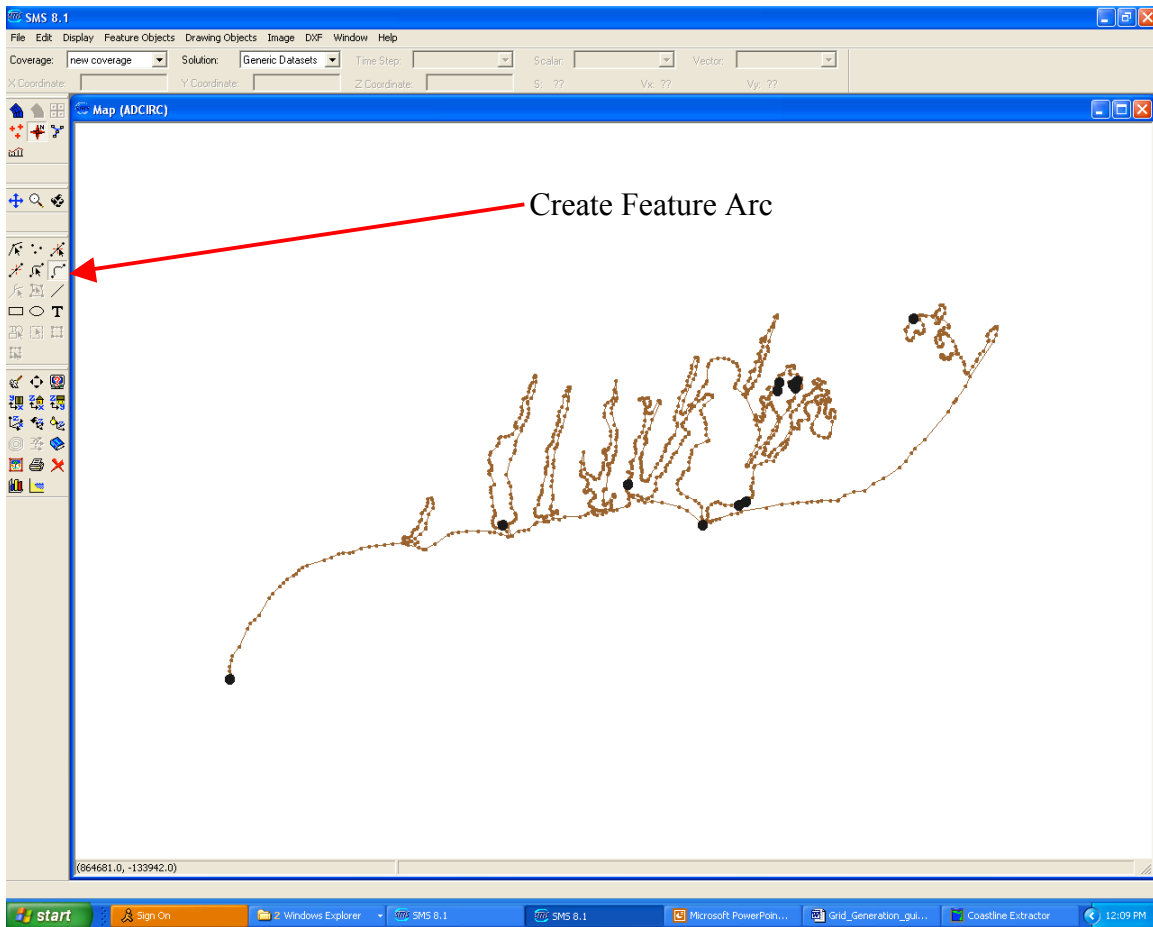
Start at a selected point at the coastline with one click, and then hold "shift", click points to build an open boundary. Be sure to double click at the end point.
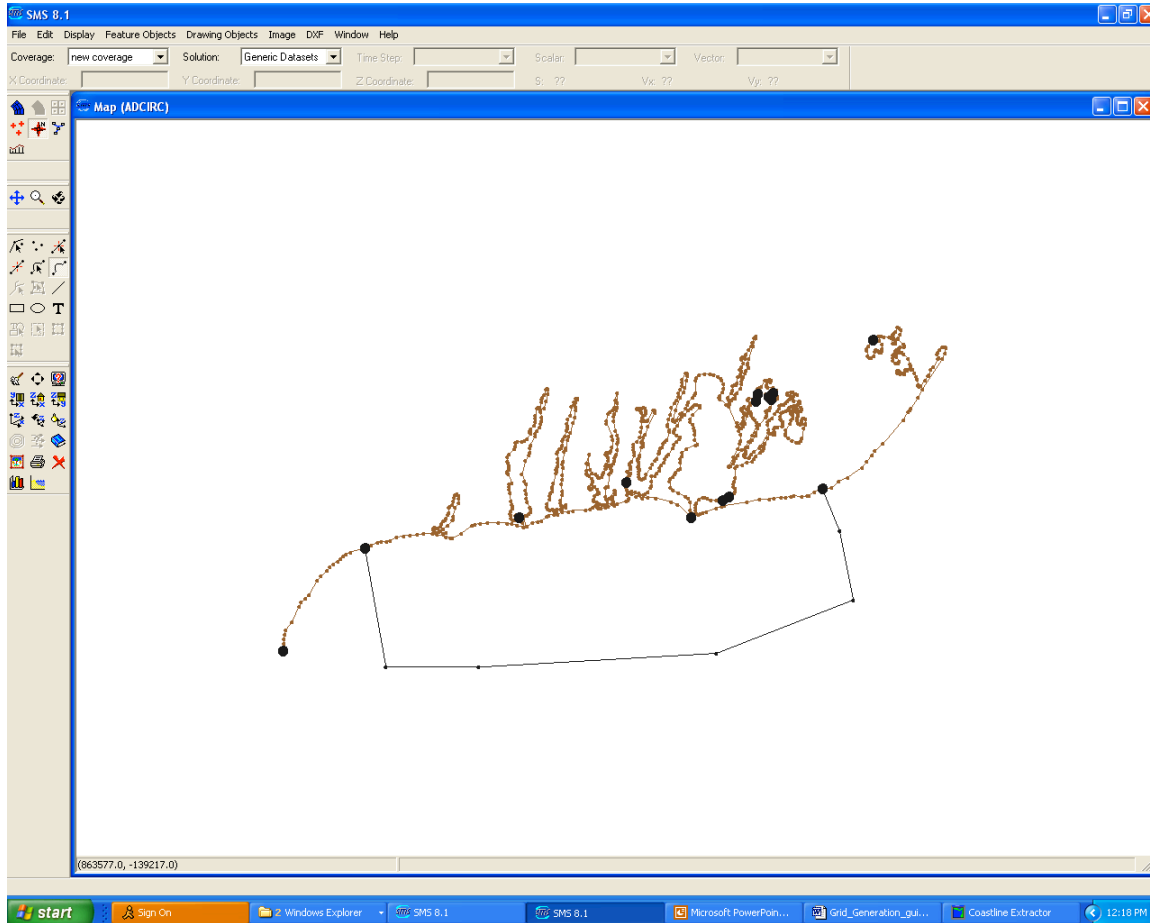
The dots are the points you click.

4. Smooth the open boundary line

Select "Select Feature Vertex", then use mouse to move the line to get the shape you like.

You can add additional points by selecting "Create Feature Vertex"
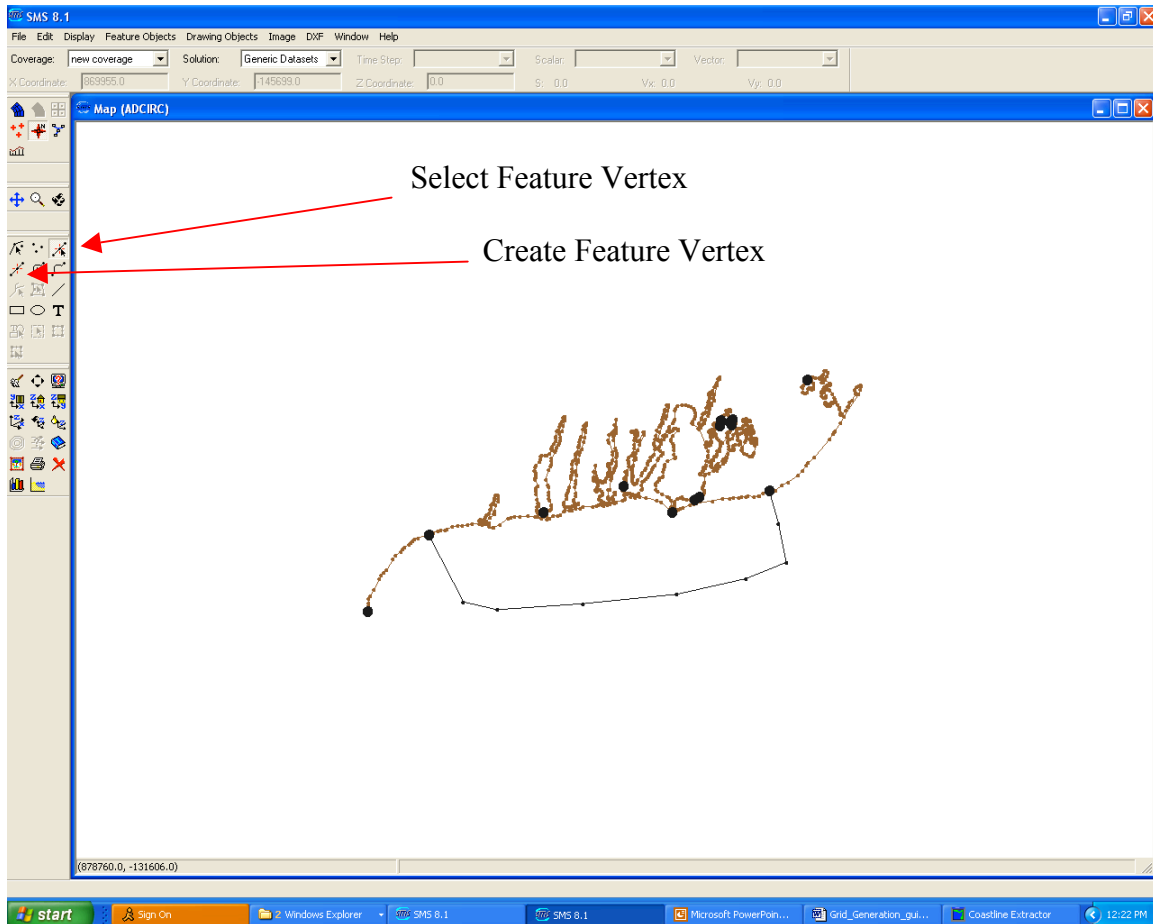
5. Choose the horizontal resolution

Click "Select Feature Arc", then move the mouse to the part of the coastline (the line with gray color is the part of the coastline you select); go to the upper manual line to select "Feature Objects" to select "Redistribute Vertices"
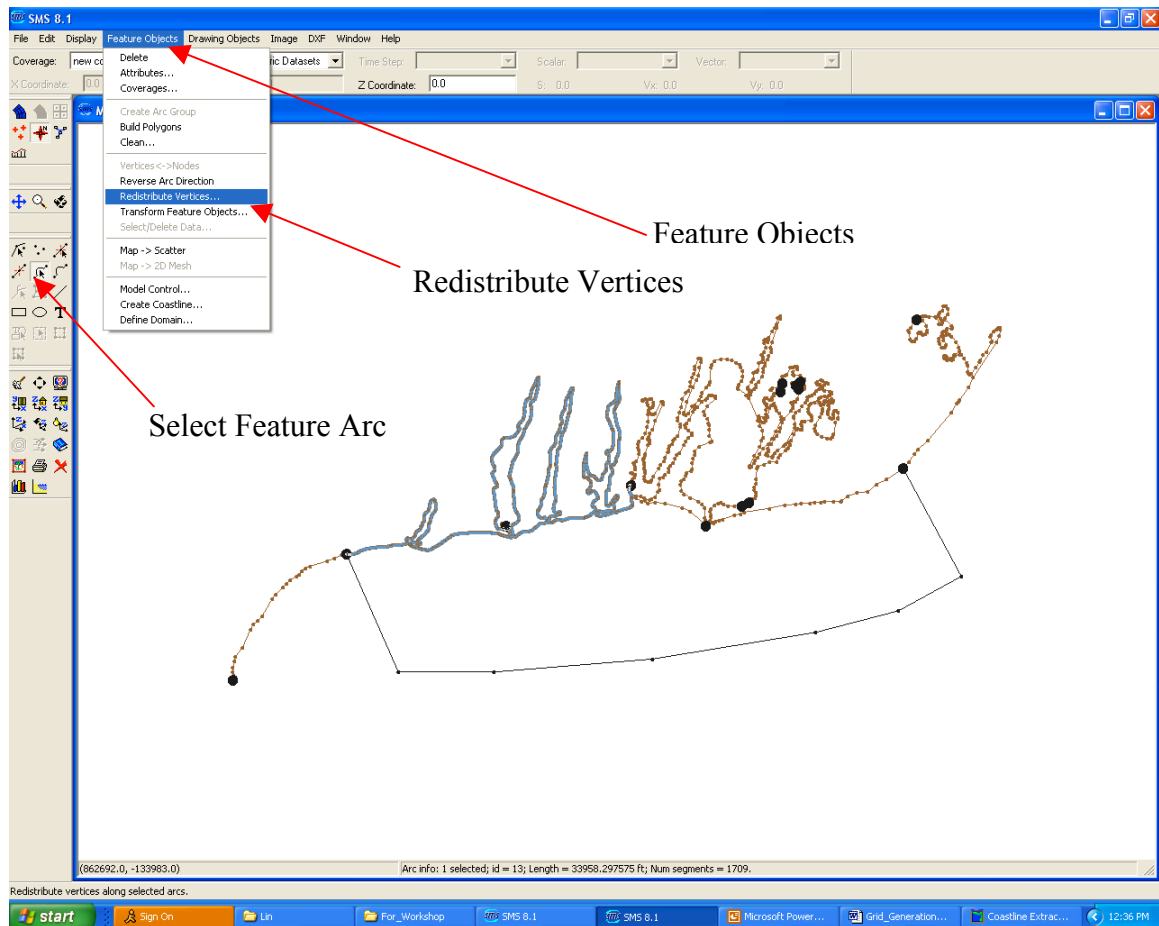
Click "Redistribute Vertices". In the sub-window of "distribute vertices", you can select the spacing interval. The unit of the length is the same as your input data. For example, we type "50" in "spacing", it means we want to have a horizontal resolution of about 50 m in the segments we choose.

Choose the spacing
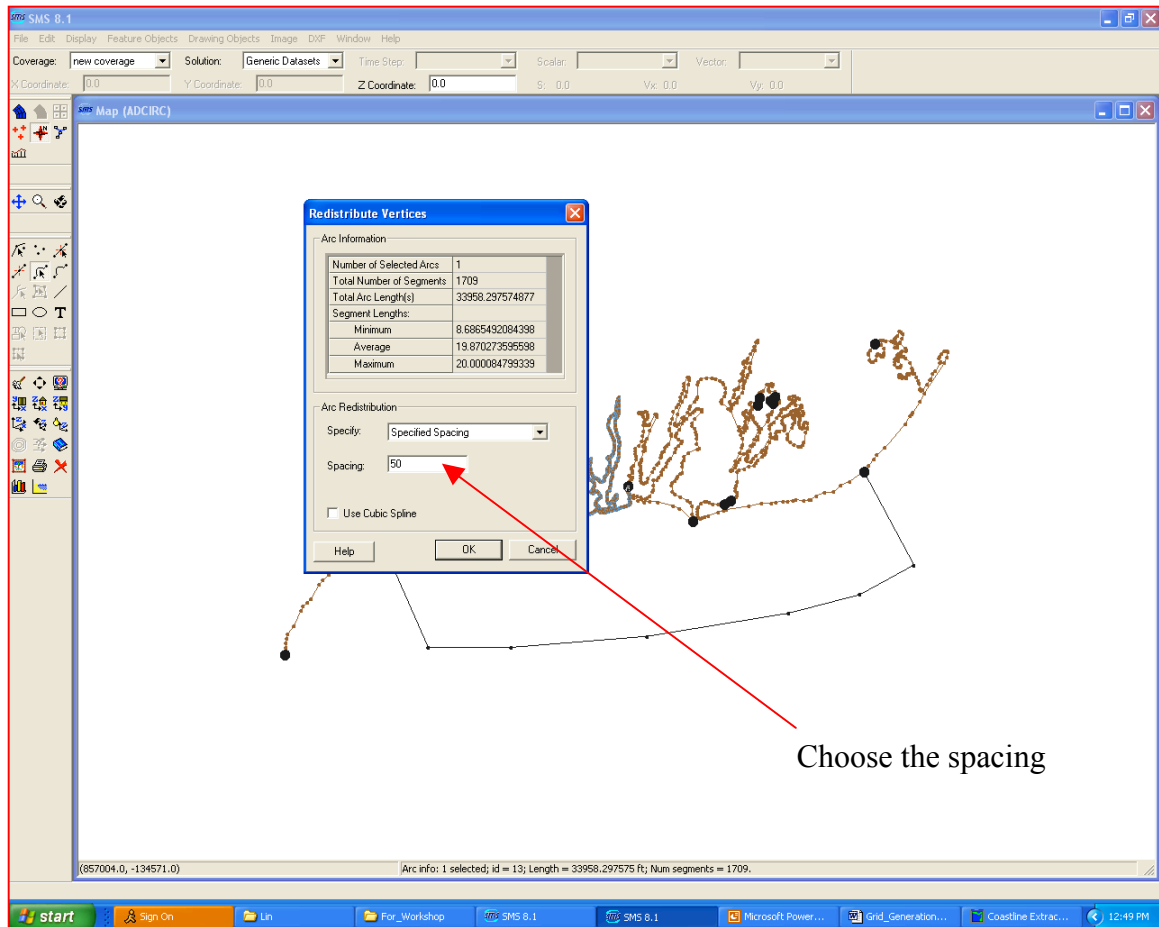
Repeat this exercise until you select all segments of the coastline and open boundary line. Be sure to save your map file. SMS does not save as your proceed and you cannot go backwards. After all these are completed, the screen should look like:
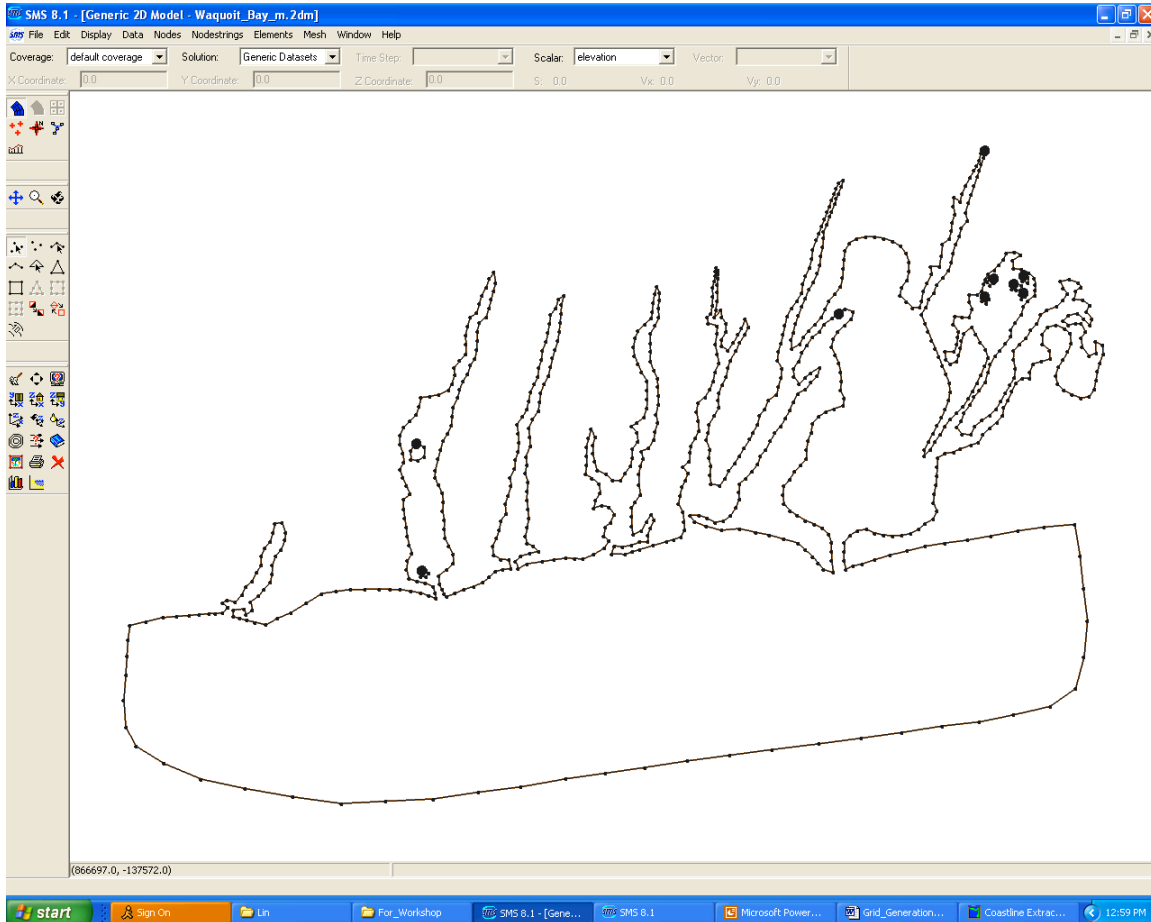
Next step is to build mesh!

6. Build the mesh

Click "Select Feature Arc". Use your mouse to select all lines in the screen you want to include in building meshes. Then go to 'Feature Objects' menu and select 'Build polygons'.
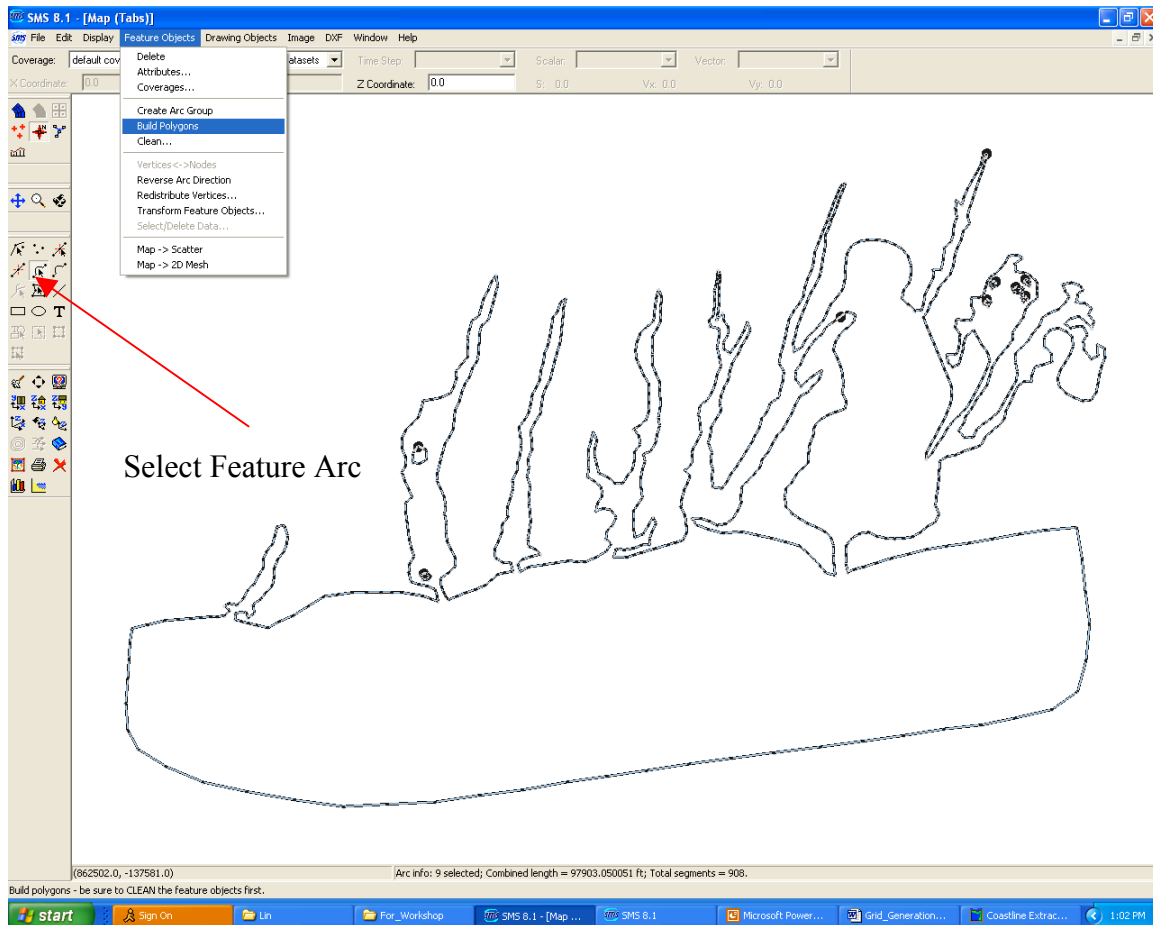
Select Feature Arc

Click "Select Feature Polygons" on the left menu bar, then move the mouse back to the mesh domain, hold the shift key, and click it, the entire domain should become black. If you find some regions are not colored black, it means that your previous steps are not done correctly, either the line is not closed or perhaps another issue. Be sure to go back to fix it. When correct, your screen should look like this:
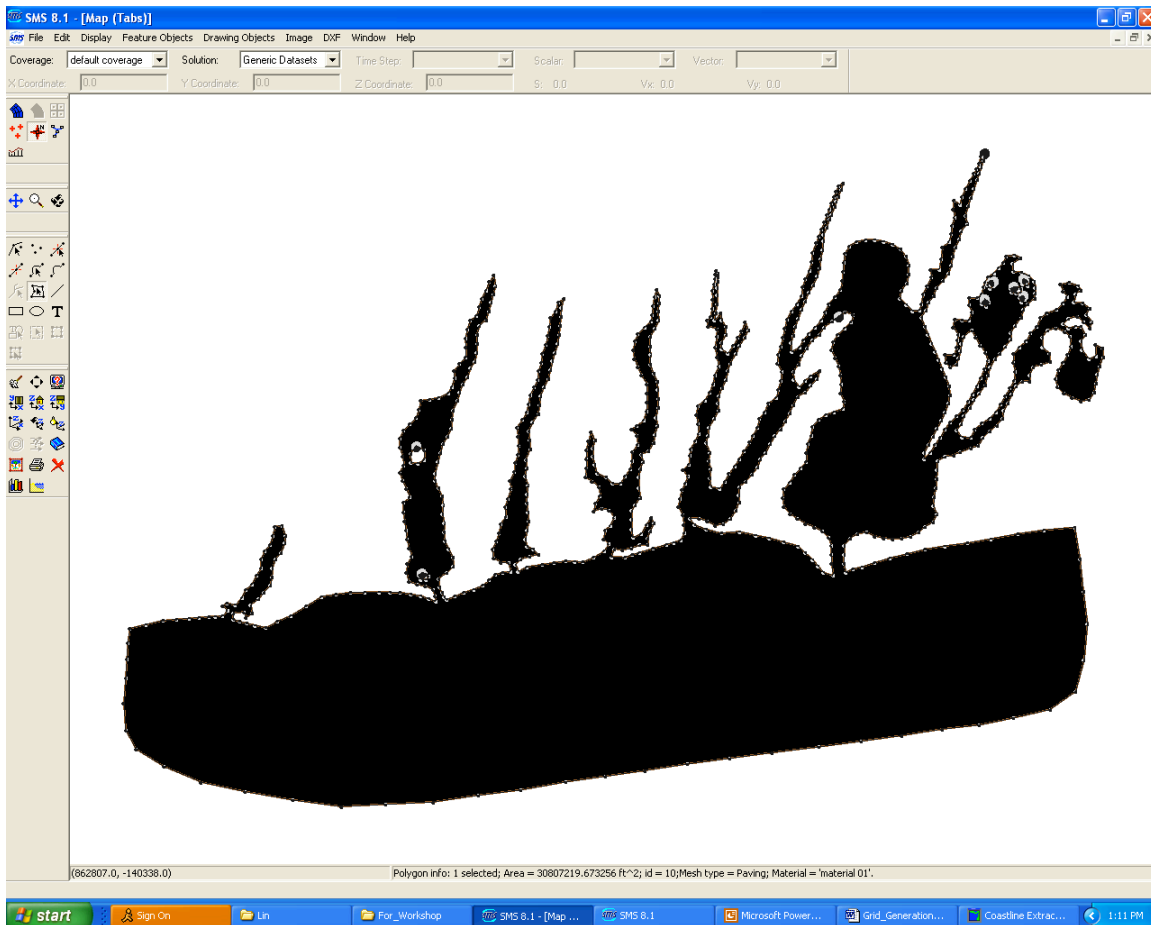
Go to the upper menu bar to and select "Map→2D Mesh" from the "Feature Objects"' pull-down menu. Click on it to start building your mesh.
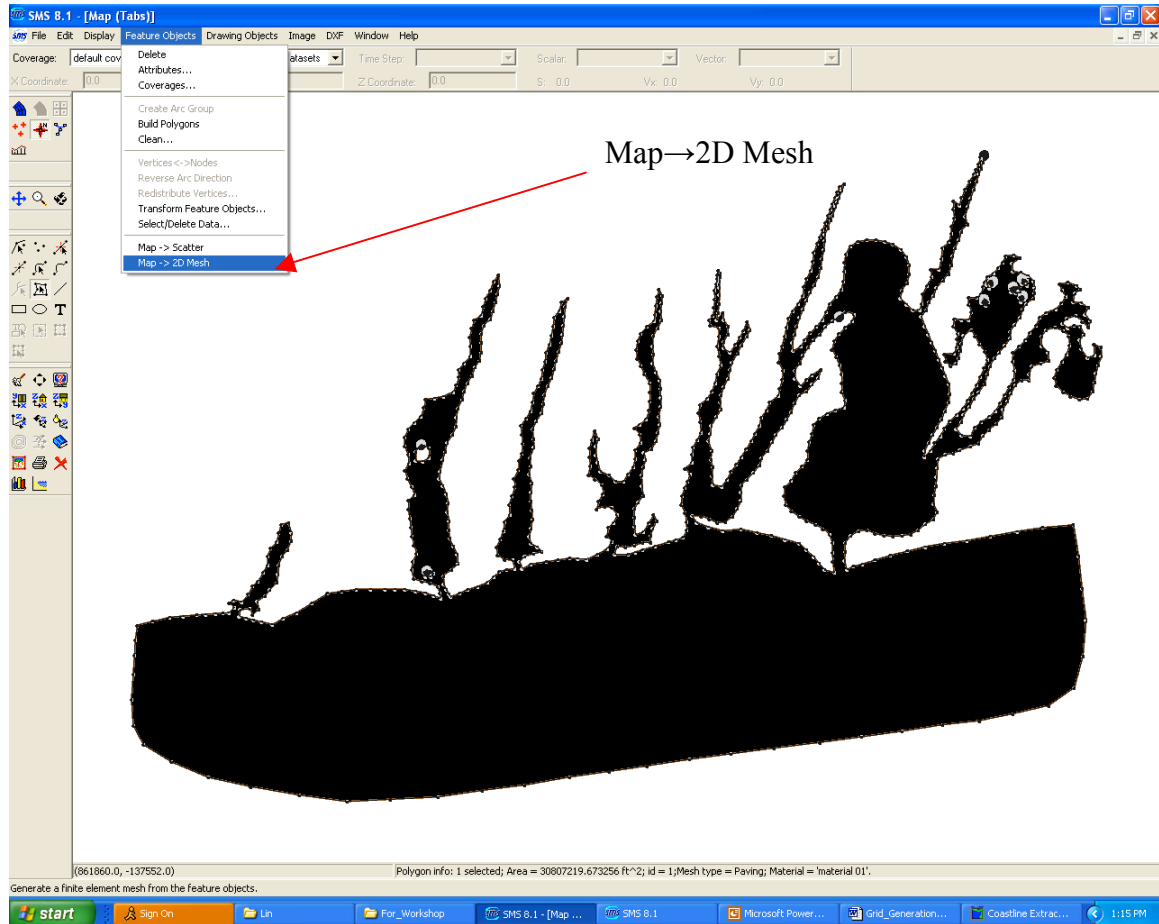
Now, the mesh is built. See below for the result.
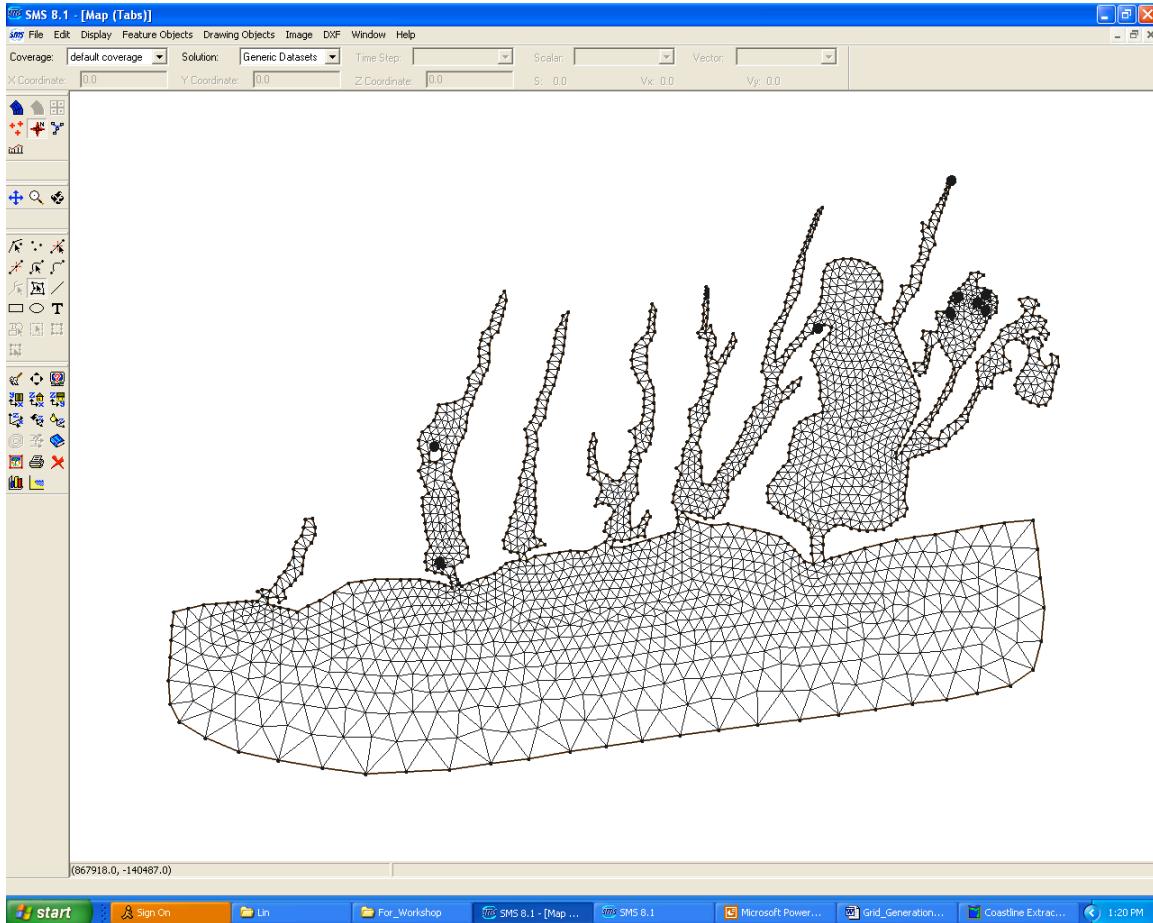
Be sure to save this mesh. First, click "Mesh Module" on the left side manual bar, then go to the upper menu "File" to select "Save Mesh". Click on it, your mesh is saved.
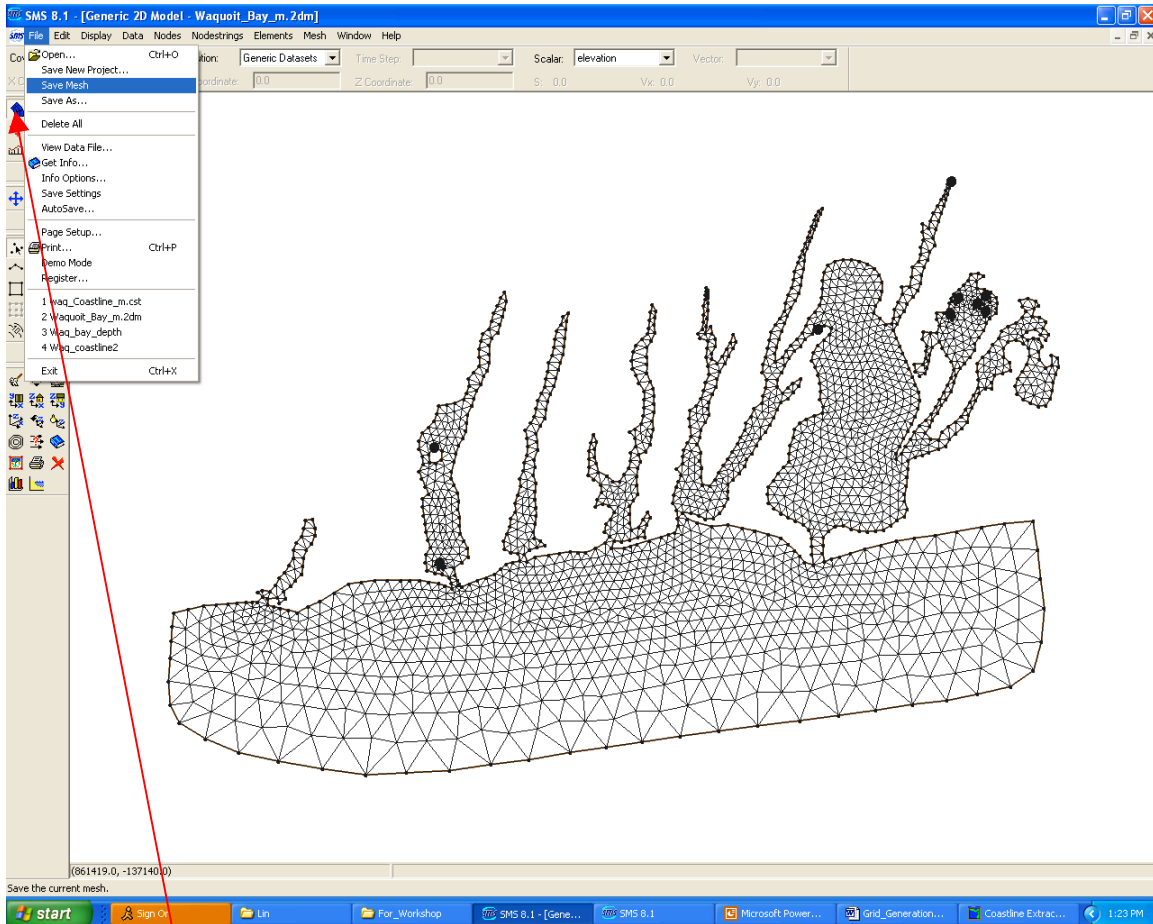
Mesh Module

7. Mesh Quality Control

Step 1: unlock the mesh. Remain in "Mesh Module", go to the upper menu to select "Nodes". In the sub-window of "Nodes", you will find "locked" box which will be checked. That means that the mesh is locked. Click "locked" to unlock the mesh.

Step 2: go to the left menu bar to select "Display Options". Click on it, you should see a screen like this:
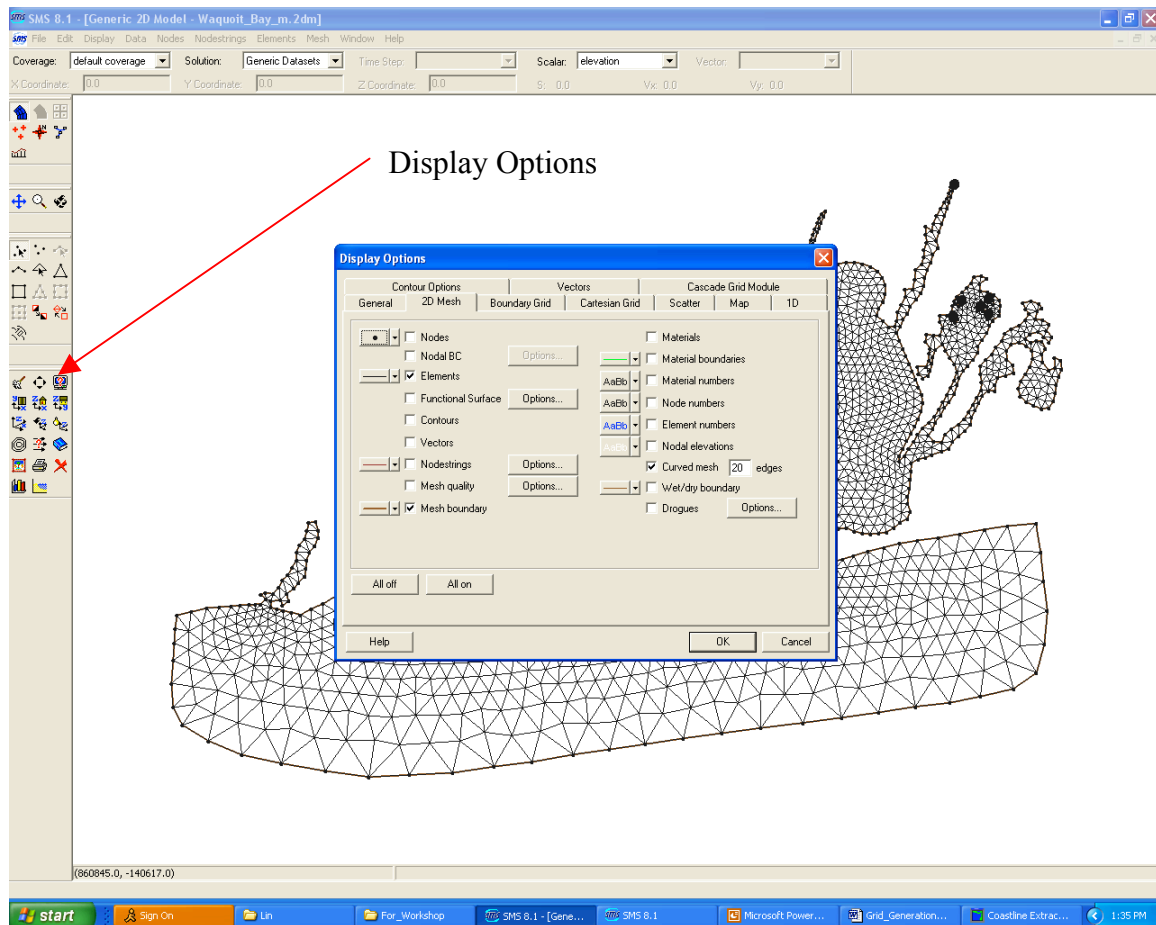
Step 3: check the box "Mesh quality", click "options" next to "Mesh quality", an element quality checks screen appears. Users need to input their quality requirement parameters into each row. Below are our recommendations:

Minimum interior angle: 30.0

Maximum interior angle: 130.0

Maximum slope: 0.1

Element area change: 0.5

Connecting elements: 8 or less (8 is the maximum elements allowed for FVCOM).

After all selections, click "ok".
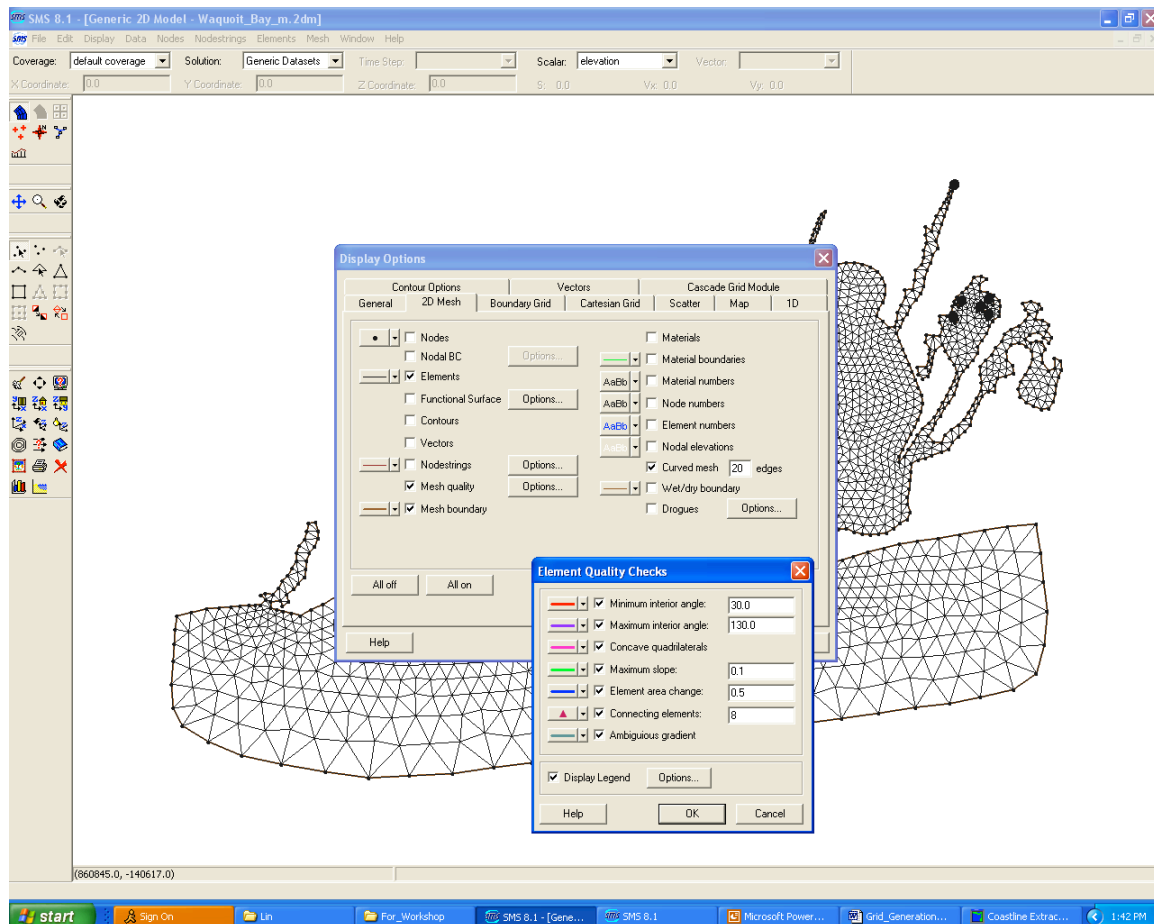
Below is the result of the quality selection. Red colors show that the angle of an element is out of the range we specified, and blue colors show that the element area change of those triangles are out of the range we expected.
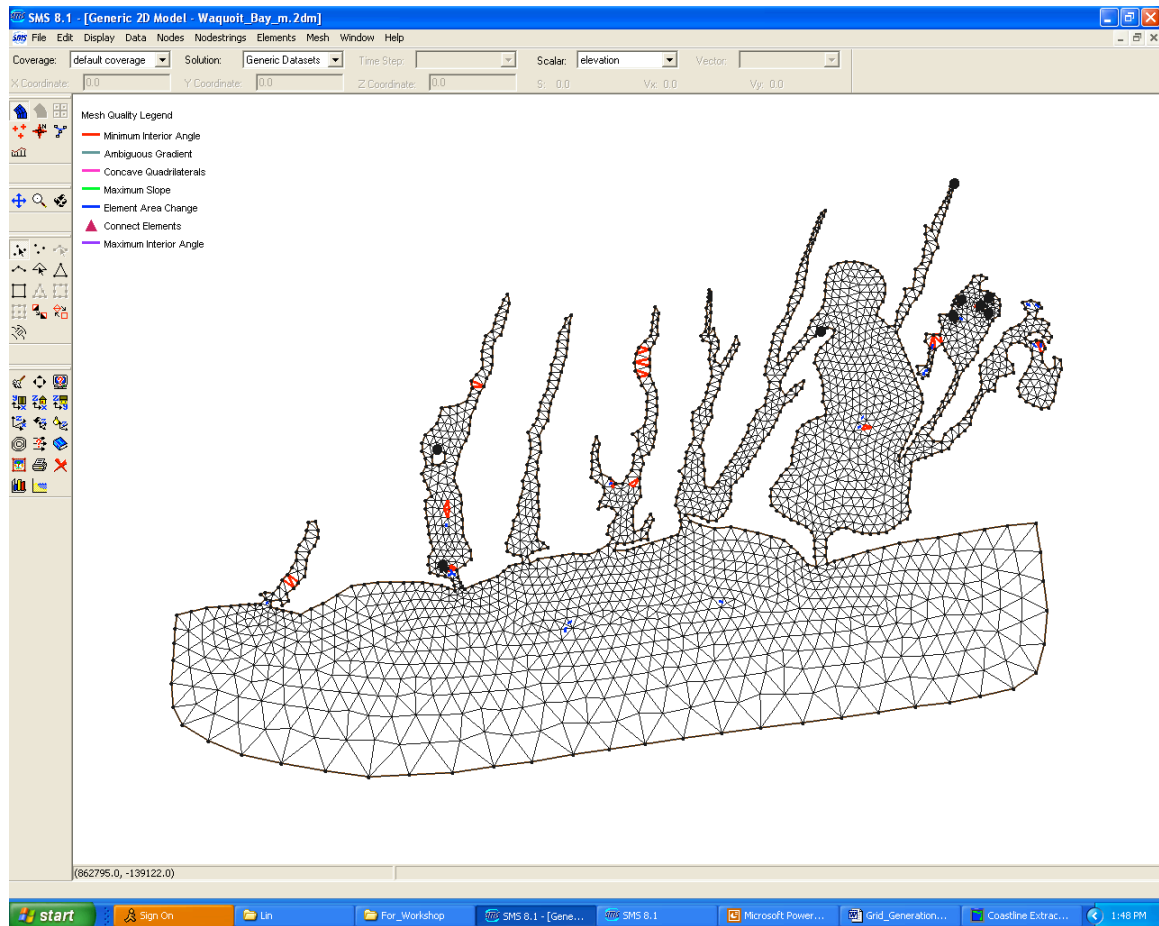
Step 4: start adjusting the mesh angles and area. The simplest way is to adjust using the mouse. Go to the left manual bar to click on "Select Mesh Node". Use the mouse to go to the colored area, and move the nodes until the color disappears. Repeat for all colored regions.
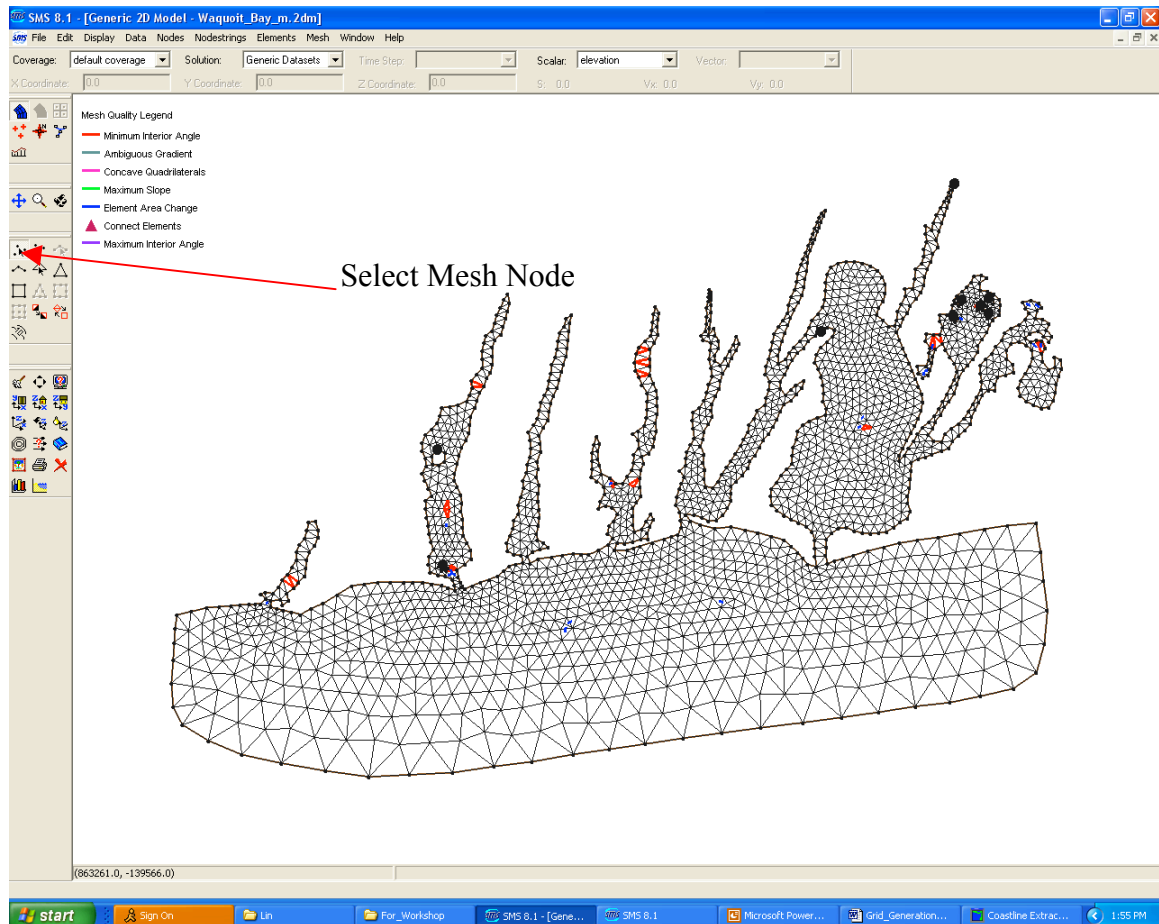
Note: To make the radiation open boundary condition work properly, we suggest that users make one of the interior edges of an open boundary triangle normal to the open boundary. FVCOM will run without this restriction, but with increased numerical noise due to the high frequency wave reflection from the open boundary.

8. Select the open boundary nodes to build an input file for the open boundary condition treatment.

Step 1: Go to the left manual bar to click "Create Nodestring". Move your mouse to the first node of the open boundary line connected to the coastline. Hold "shift" and then continue to click the open boundary nodes. When you get to the last end node point connected to the other side of the coastline, double click it. Check the screen to make sure all boundary nodes are selected.
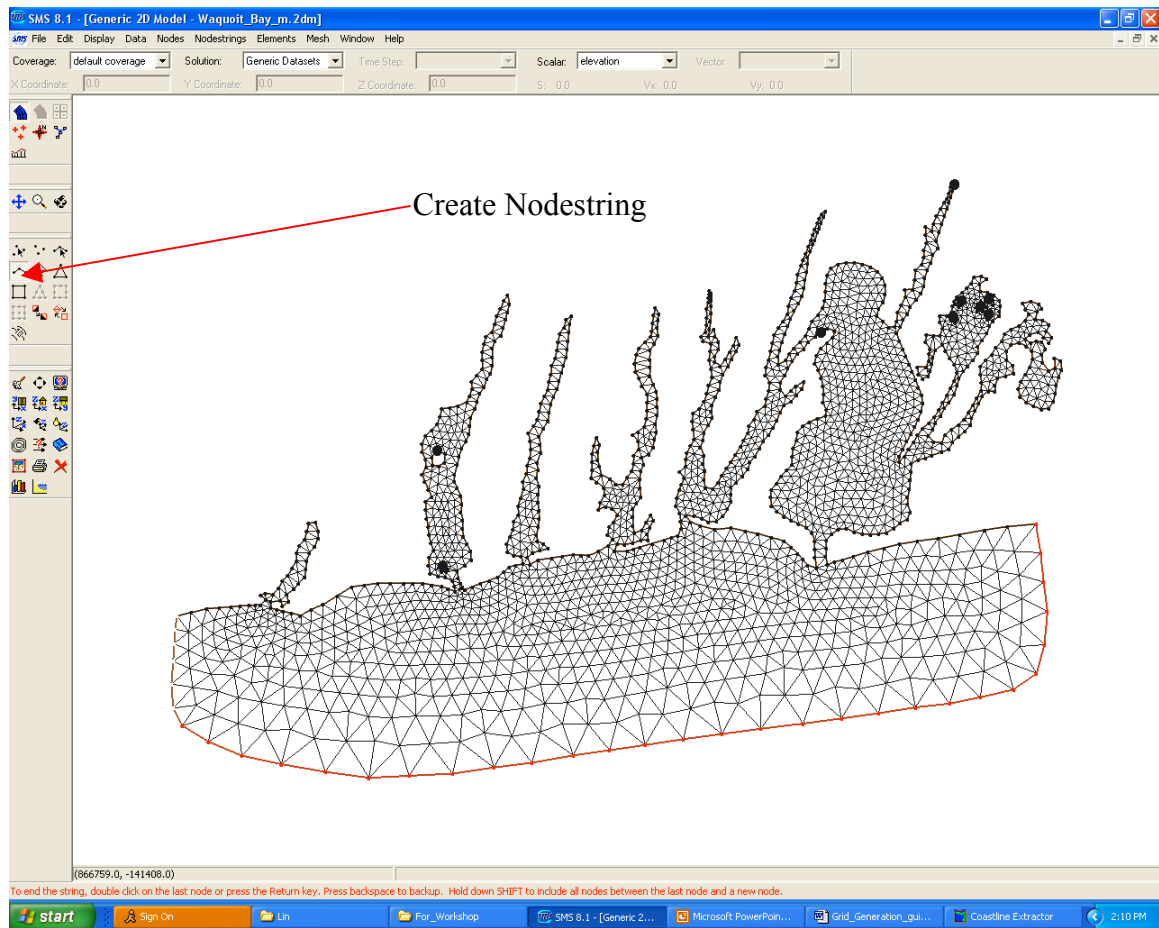
Step 2: Re-number the open boundary nodes. Go to the left manual bar to click on "Select Nodestring". A small box will appear on the open boundary line. Click this box, all the open boundary nodes will be colored. Go to the upper manual bar to select "Nodestring". On that sub-window, click on "renumber". All nodes on the open boundary will be renumbered.
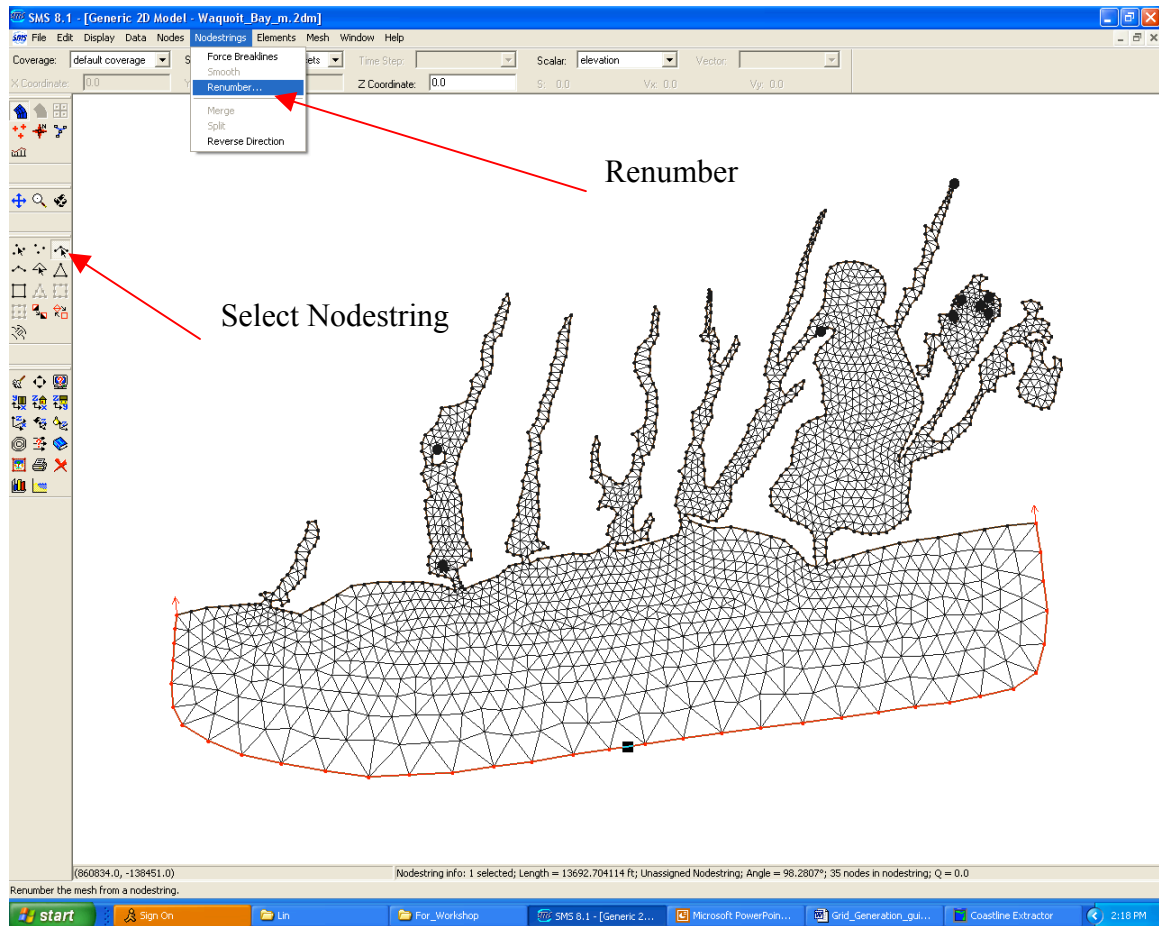
After you have finished this step, be sure to go to "File" to click "**Save Mesh**", so that the mesh file will be updated with the new numbering of the open boundary nodes. This numbering will facilitate specifying the open boundary forcing in later model setup stages.

9. Interpolation of bathymetric data into the mesh

Step 1: Go to the upper manual to "File" and select "Open". Find the bathymetric file in your computer, then click it. Keep clicking "next" in the GUI selection. All the data will appear on the screen, overlapping over the mesh. An example is given below. Note: this is just used to demonstrate how we can interpolate the bathymetric data into the mesh, so we did not use the complete bathymetric data set here.
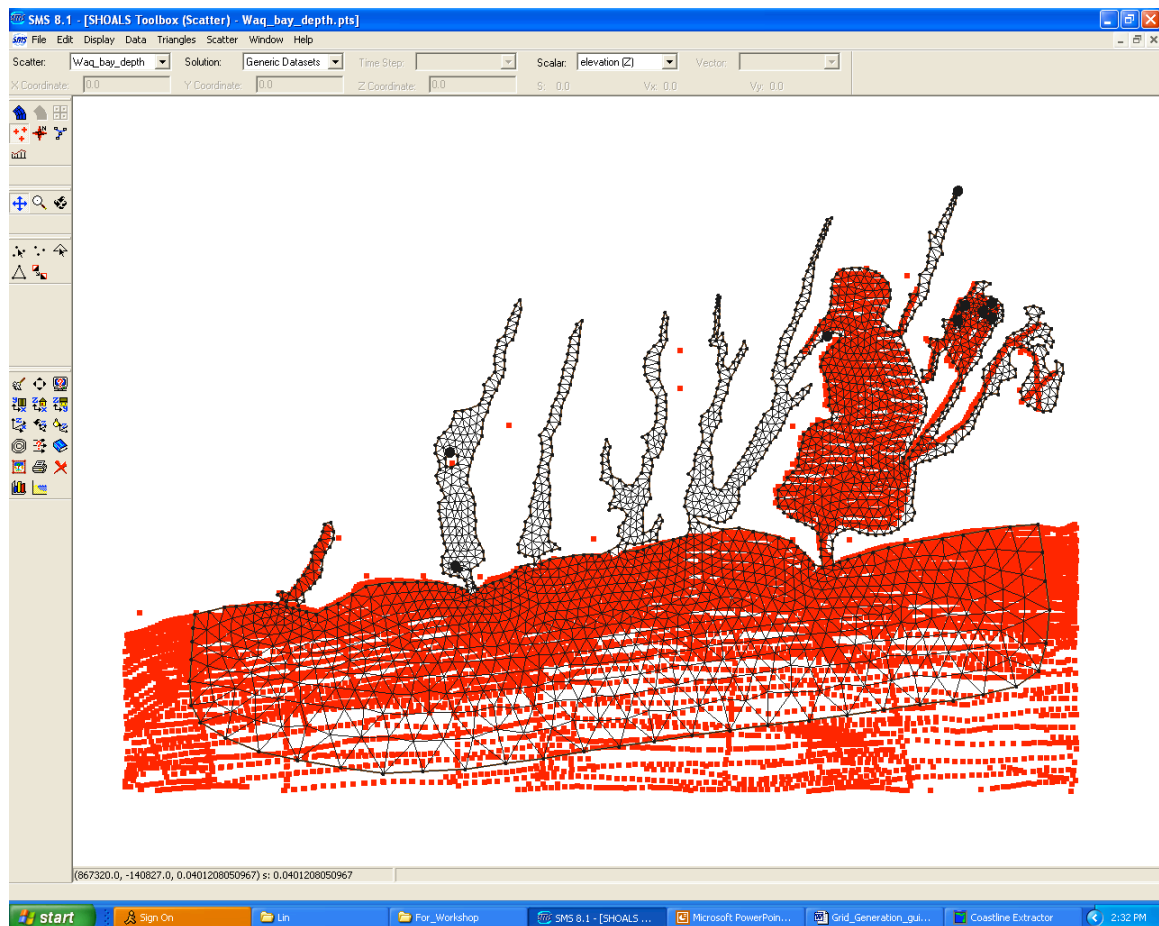
Step 2: Go to the upper manual bar, select "Scatter". In the sub-window manual, select "Interpolate to Mesh". See the screen below for example.
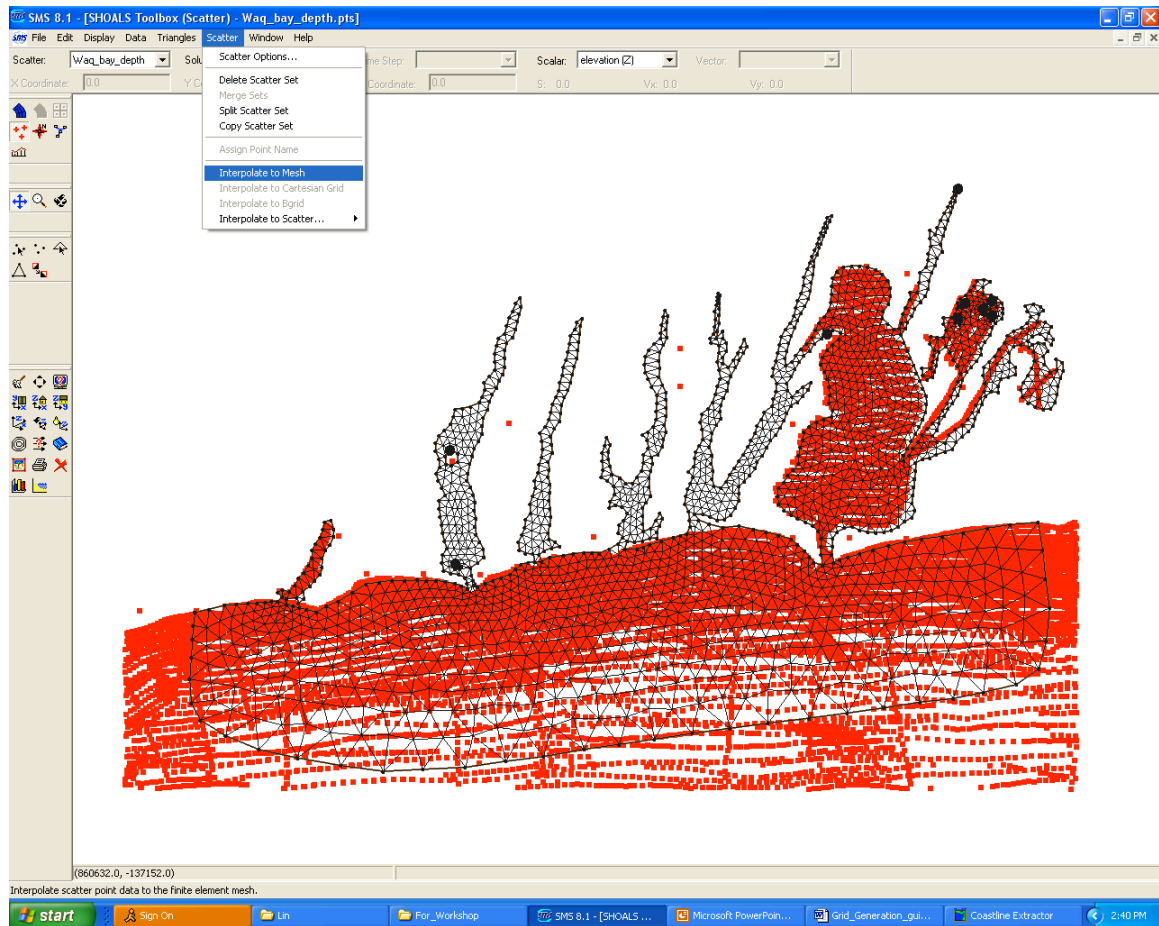
Step 3: Click "Interpolate to Mesh" You can select different interpolation methods in the box of "interpolation". After your make your selection, click "ok". The bathymetric data will be interpolated to nodes of the mesh.
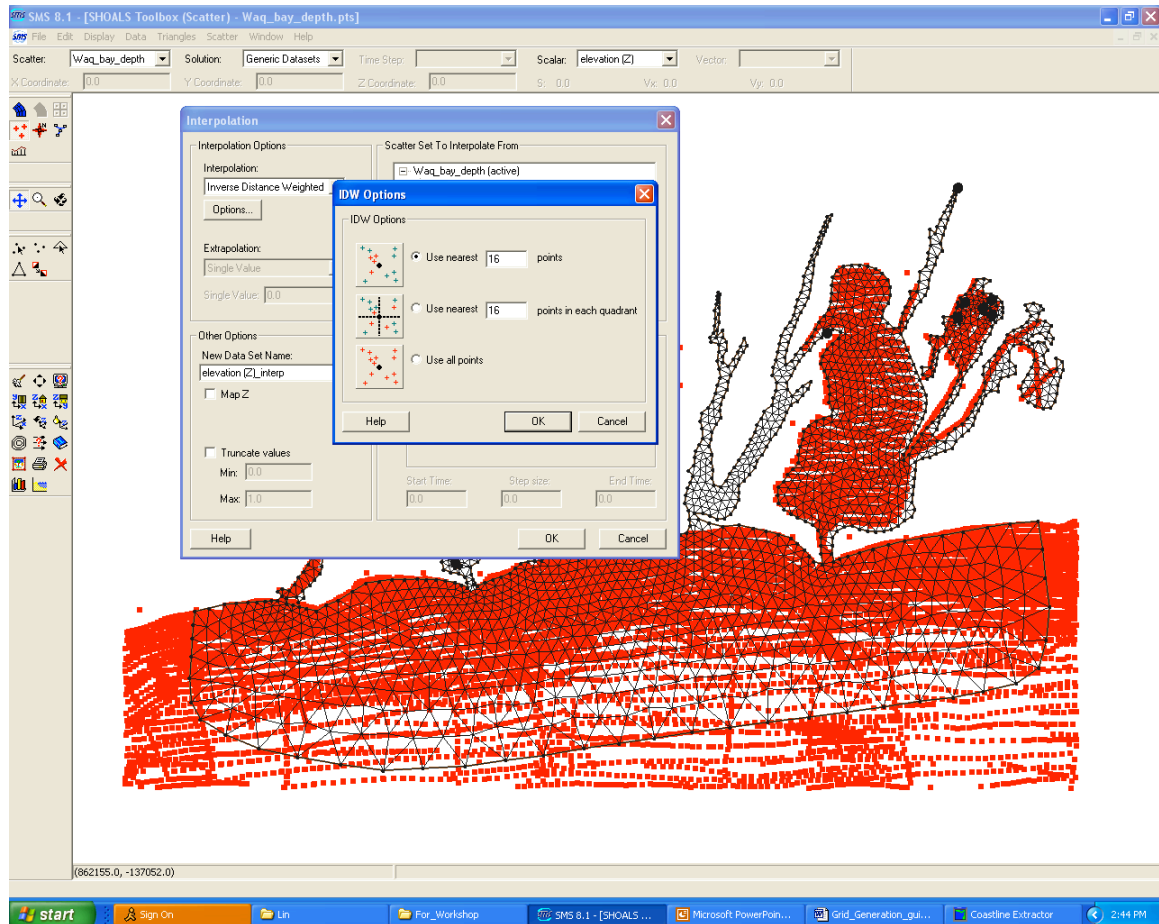
Step 4: Select "save as" from the file menu and create an output file name with the extension "pts" (example: myfile.pts). Click "Save", a sub-window titled "Export Shoal File" screen will appear. From here, you must do two things: 1) in "Export Option" check "Save Mesh"; and 2) go to "Column Header", select "Z" row and then choose the last option "elevation (Z) inter(Scalar)". After you complete these two steps, click "ok" and your bathymetric data file is saved.
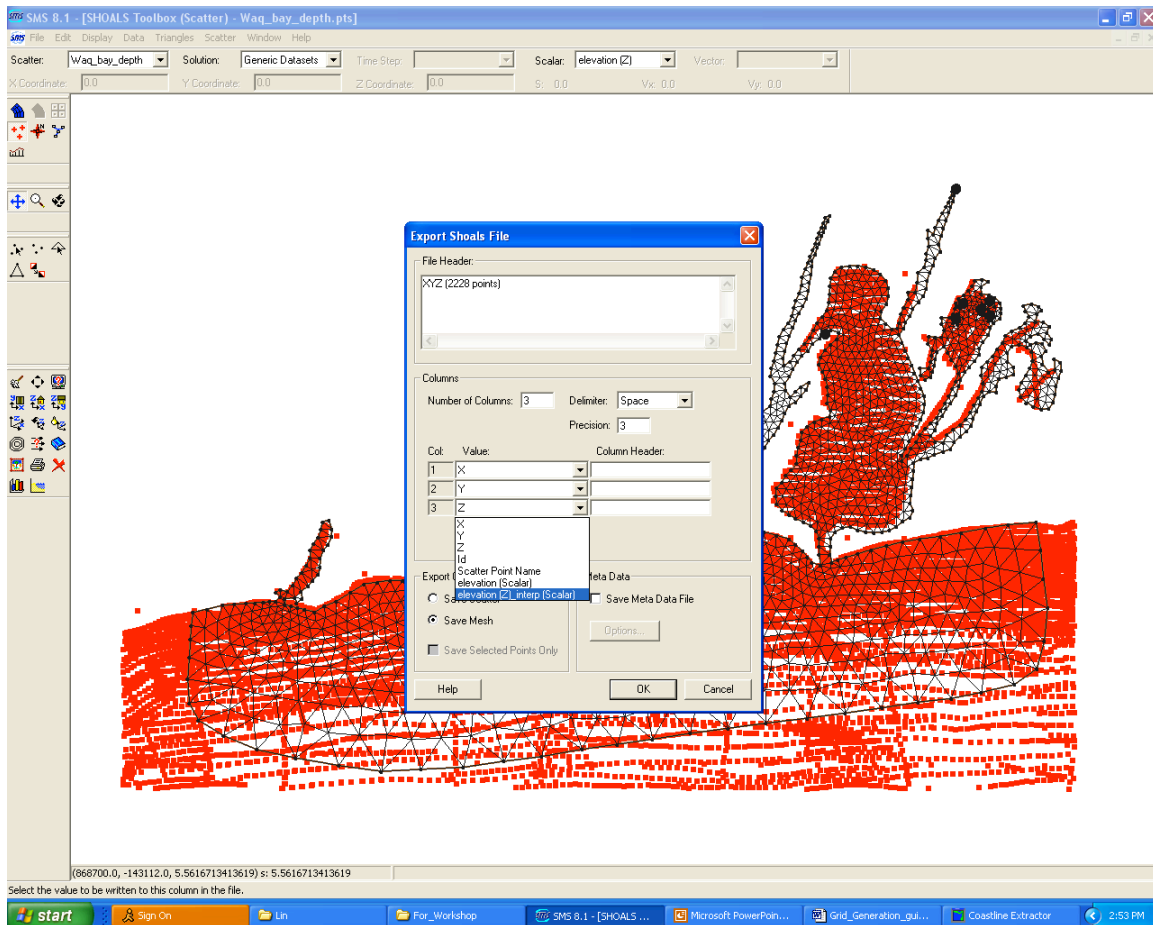
10. Displaying the Mesh and Interpolated Bathymetry

Users can go to "Display Options" on the left manual bar to see different methods of displaying the mesh and interpolated bathymetry. Shown below is an example of the bathymetry image and mesh produced by selecting "2D Mesh" and "Contour Options".

Acknowledgements

References

Ambrose Jr, R. B., T. A. Wool, and J. L. Martin, 1993. The water quality analysis simulation program, WASP5, Part A: Model documentation. U. S. Environmental Protection Agency, Athens, Georgia, 202 pp.

Armfield, S. and R. Street, 2002. An analysis and comparison of the time accuracy of fractional-step methods for the Navier–Stokes equations on staggered grids. *Int. J. Numer. Methods Fluids,* 38, 255–282.

Armstrong, R. A., 1994. Grazing limitation and nutrient limitation in marine ecosystems: steady state solutions of an ecosystem model with multiple food chains. *Limnol&Oceanogr.*, 39(3), 597-608.

Anderson, J. L., 2001. An ensemble adjustment Kalman filter for data assimilation. *Mon Weather Rev.*, 129: 2884–2903.

Anderson, B. and J. B. Moore, 1979. *Optimal Filtering*, Prentice-Hall, 357pp..

Anderson, L. A., A. R. Robinson and C. J. Lozano, 2000. Physical and biological modeling in the Gulf Stream region: I. Data assimilation methodology. *Deep-Sea Res.* I, 47, 1787-1827.

Anderson, V. and P. Nival, 1991. A model of the diel vertical migration of zooplankton based on euphausiids. *J. Mar. Res.*, 49: 153-175.

Anderson, T. R. and P. J. le B. Williams, 1999. A one-dimensional model of dissolved organic carbon cycling in the water column incorporating combined biological-photochemical decomposition. *Global Biogeochemical Cycles,* 13, 337-349, doi:10.1029/1999 GB900013.

Austria, P. M. and A. A. Aldama. 1990. Adaptive mesh scheme for free surface flows with moving boundaries. In G. Gambolati, A. Rinaldo, C. A. Brebbiam, W. G. Fray, and G. F. Pinder (eds.), Computational methods in Surface Hydrology, Springer-Verlag, New York, p. 456-460.

Baly, E. C. C., 1935. The kinetics of photosynthesis. *Proc. R. Soc. Lond. Ser. B*, 117, 218-239.

Bannister, T. T., 1979. Quantitative description of steady state, nutrient-saturated algal growth, including adaptation. *Limnol&Oceanogr.*, 24, 76-96.

Barlay,S., K. Buschelman, V. Eijkhout, W. Group, D. Kaushik, M. Knepley, L. G. McInnes, B. Smiith and H. Zhang, 2007. PETSc user manual. ANL-95/11-Revision 2.3.3., *Mathematics and Computer Science Division*, Argonne National Laboratory, 190pp.

Barth, T. J., 1993. Recent development in high order K-exact reconstruction on unstructured meshes. *AIAA paper 93-0668*.

Beardsley, R. C. and D. B., Haidvogel, 1981. Model studies of the wind-driven transient circulation in the Middle Atlantic Bight. Part 1: Adiabatic boundary conditions. *J. Phys. Oceanogr.*, 11, 355-375.

Bergamasco, A., P. Malanotte-Rizzoli, W. C. Thacker and R. B. Long, 1993. The seasonal steady circulation of the Eastern Mediterranean determined with the adjoint method. *Deep-Sea Res.*, 40, 1269–1298.

Bentzen, E., W. D. Taylor and E. S. Millard, 1992. The importance of dissolved organic phosphorus to phosphorus uptake by limnetic plankton. *Limnol.&Oceanogr.*, 37, 217-231.

Bieman, V. J. and D. M. Dolan, 1981. Modeling of phytoplankton-nutrient dynamics in Saginaw bay, Lake Huron. *J. Great Lake Res.*, 7(4), 409-439.

Bishop, C. H., B. J. Etherton and S. J. Majumdar, 2001. Adaptive sampling with the Ensemble Transform Kalman Filter. Part I: theoretical aspects, *Mon.Wea,Rev.,* 129, 420-436.

Bissett, W.P., J.J. Walsh, D.A. Dieterle and K.L. Carter 1999. Carbon cycling in the upper waters of the Sargasso Sea: I. Numerical simulation of differential carbon and nitrogen fluxes. *Deep-Sea Res. I*, 46, 205-269.

Blanchet, I., C. Frankignoul and M. A. Cane, 1997. A comparison of adaptive Kalman filter for a Tropical Pacific Ocean model. *Mon. Wea. Rev.*, 125, 40–58

Blumberg, A. F. and L. H. Kantha, 1985. Open boundary condition for circulation models. *J. Hydraul. Eng.*, 111, 237-255.

Blumberg, A. F. and G. L. Mellor, 1987. A description of a three-dimensional coastal ocean circulation model. In Three-dimensional Coastal Ocean Model, N.S. Heaps, Ed*., Coastal. Estuar. Sci.*, 4, 1-6.

Blumberg, A. F., 1994. A primer for Ecom-si. *Technical Report of HydroQual, Inc.*, 66 pp.

Booij, N., R. C. Ris and L. H. Holthuijsen, 1999. A third-generation wave model for coastal regions, Part I, Model description and validation. *J. Geophys. Res.*, 104, C4, 7649-7666.

Boris, J.P. and D. L. Book, 1973. Flux corrected transport I, SHASTA, a fluid transport algorithm that works. *J. Comp. Phys.* 11, 38-69.

Briegleb, B. P., 1992. Longwave band model for thermal radiation in climate studies. *J. Geophys. Res.*, 97, 11475–11485.

Bryant, A. D., M. Heath, W. S. G. Gurney, D. J. Beare and W. Robertson, 1997. The seasonal dynamics of *Calanus finmarchicus*: development of a three-dimensional structured population model and application to the northern North Sea. *J. Sea Res.* 38, 361-379.

Buehner, M. and P. Malanotte-Rizzoli, 2003. Reduced-rank Kalman filters applied to an idealized model of the wind-driven ocean circulation, *J. Geophys. Res.*, 108, 3192, doi:10.1029/2001JC000873

Burchard, H., 2001. Simulating the wave-enhanced layer under breaking surface waves with two-equation turbulence models. *J. Phys. Oceanogr.*, 31, 3133-3145.

Burchard, H., 2002. Applied turbulence modeling in marine waters. *Springer:Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan Paris-Tokyo*, 215pp.

Burchard, H. and H. Baumert, 1995. On the performance of a mixed-layer model based on the $k-\varepsilon$ turbulence closure. J. Geophys. Res., 100 (C5), 8523-8540.

Canuto, V. M., A. Howard, Y. Cheng and M. S. Dubovikov, 2001. Ocean turbulence. Part I: one point closure model-momentum and heat vertical diffusivities. *J. Phys. Oceanogr.*, 31, 1413-1246.

Casulli, V., 1999. A semi-implicit finite-difference method for non-hydrostatic free-surface flows. *Int. J. Numer. Methods Fluids,* 30, 425–440.

Casulli, V. and E. Cattani. 1994. Stability, accuracy and efficiency of a semi-implicit method for three-dimensional shallow water flow. *Computers and Mathematics with Application*, 27: 99-112.

Casulli, V. and G. S. Stelling, 1998. Numerical simulation of 3D quasi-hydrostatic, free-surface flows. *J. Hydraulic Eng. (ASCE)*, 124:678－686

Caperon, J. and J. Meyer 1972. Nitrogen-limited growth of marine phytoplankton II. Uptake kinetics and their role in nutrient limited growth of phytoplankton. *Deep-Sea Res.* 19, 619-632.

Casulli, V. and R. T. Cheng. 1991. A semi-implicit finite-difference model for three-dimensional tidal circulation. In M. Spaulding et al. (eds.), *Proceeding of 2nd International Conference on Estuarine and Coastal Modeling*, ASCE, Tampa, Florida, p. 620-631.

Casulli, V. and R. T. Cheng. 1992. Semi-implicit finite-difference methods for three-dimensional shallow-water flow. *International Journal for Numerical Methods in Fluids*, 15, 629-648.

Cerco, C. F. and T. M. Cole, 1993. Three-dimensional eutrophication model of Chesapeake Bay, *ASCE /. Environ. Engr. Div.*, 119(6), 1006-1025.

Cerco, C. F., B. Bunch, M. Dortch, B. Johnson and K. Kim, 2003. Eutrophication and pathogen abatement in the San Juan Bay Estuary. *J. Eniv. Eng.*, 129(4), 318-327.

Cerco, C. F. and M. R. Noel, 2004. Process-based primary production modeling in Chesapeake Bay. *Mar. Ecol. Prog. Ser.*, 282, 45-58.

Cerco, C. F., M. Noel and S-C Kim, 2006. Three-dimensional management model for Lake Washington: (II) eutrophication modeling and skill assessment. *J. Lake. Res. Management*, 22(2), 115-131.

Chapman, S. and R. S. Lindzen, 1970. Atmospheric tides: thermal and gravitational. *Goldon & Breach Publishers Hagan*, 200pp.

Chapman, D. C., 1985. Numerical treatment of cross-shelf open boundaries in a barotropic coastal ocean model. *J. Phys. Oceanogr.*, 15, 1060-1075.

Chen, C., 1992. Variability of currents in Great South Channel and over Georges Bank. Ph.D Thesis, MIT/WHOI Joint Program, Technical Report WHOI-92-20, 283pp.

Chen, C., D. Wiesenburg, and L. Xie. 1997. Influences of river discharges on biological production over the inner shelf: a coupled biological and physical model of the Louisiana-Texas shelf. *J. Mar. Res.*, 55, 293-320.

Chen, C. R. Ji, L. Zheng, M. Zhu, and M. Rawson, 1999. Influences of physical processes on ecosystem in Jiaozhou Bay: A coupled physical and biological model experiment. *J. Geophys. Res.*, 104 (C12), 29,925-29,949.

Chen, C., J. Zhu, E. Ralph, S. A. Green, and J. Budd, 2001. Prognostic modeling studies of the Keweenaw current in Lake Superior. Part I: formation and evolution. *J. Phys. Oceanogr.*, 31, 379-395.

Chen, C. R. Ji, D. Schwab, D. Beletsky, D. Fahanenstial, M. Jiang, T. H. Johengen, H. Lavrentyev, B. Eadie, J. W. Budd, M. Bundy, W. Gardner, J. Cotner, and P. J. Lavrentyer, 2002. A coupled biological and physical model study of the ecosystem in Lake Michigan Part I: A 1-D experiment. *Ecological Modeling*, 152, 145-168.

Chen, C. H. Liu, R. C. Beardsley, 2003a. An unstructured, finite-volume, three-dimensional, primitive equation ocean model: application to coastal ocean and estuaries. *J. Atm. &Oceanic Tech.*, 20, 159-186.

Chen, C, R. C. Beardsley, and P. J. S. Franks, 2003b. J. V. Keuren, Influences of the diurnally varying heat flux on stratification and residual circulation on Georges Bank. *J. Geophys. Res.*, 108(C11), 8008, DOI 10.1029/2001JC001245.

Chen, C. J. Zhu, L. Zheng, E. Ralph and J. W. Budd, 2004a. A non-orthogonal primitive equation coastal ocean circulation model: application to Lake Superior. *J. Great Lakes Res.*,30, 41-54.

Chen, C, G. Cowles and R. C. Beardsley, 2004b. An unstructured grid, finite-volume coastal ocean model: FVCOM User Manual. SMAST/UMASSD Technical Report-04-0601, pp183.

Chen, C, R. C. Beardsley, S. Hu, Q. Xu, and H. Lin, 2005. Using MM5 to hindcast the ocean surface forcing fields over the Gulf of Maine and Georges Bank region. Journal of Atmospheric and Oceanic Technology, 22(2), 131-145.

Chen, C., R. C. Beardsley, H. Huang. J. Qi, G. Cowles, G. Gao and H. Lin, 2006a. A spherical coordinate version of FVCOM: validation and application. Unpublished manuscript.

Chen, C., J. Qi, R. C. Beardsley, H. Liu, H. Lin, and G. Cowles, 2006b. A 3-dimensional, unstructured grid, finite-volume wet/dry point treatment method for FVCOM. Unpublished manuscript.

Chen, C, R. C. Beardsley and G. Cowles, 2006c. An unstructured grid, finite-volume coastal ocean model (FVCOM) system. Special Issue entitled "Advance in Computational Oceanography", *Oceanography*, 19(1), 78-89.

Chen, C., G. Cowles and R. C. Beardsley, 2006d. An unstructured grid, finite-volume coastal ocean model: FVCOM User Manual. Second Edition. SMAST/UMASSD Technical Report-06-0602, pp 245.

Chen, C., H. Huang, R. C. Beardsley, H. Liu, Q. Xu, and G. Cowles, 2007. A finite-volume numerical approach for coastal ocean circulation studies: comparisons with finite difference models. *J. Geophys. Res*. 112, C03018, doi:10.1029/2006JC003485.

Chen, C, G. Gao, J. Qi, A. Proshutinsky, R. C. Beardsley, Z. Kowalik, H. Lin and G. Cowles, 2009a. A new high-resolution unstructured-grid finite-volume Arctic Ocean model (AO-FVCOM): an application for tidal studies. *J. Geophys. Res*., doi: 10.1029/2008jc004941.

Chen, C., P. Malanotte-Rizzoli, J. Wei, R. C. Beardsley, Z. Lai, P. Xue, S. Lyu, Q. Xu, J. Qi, and G. W. Cowles, 2009b. Application and comparison of Kalman filters for coastal ocean

problems: An experiment with FVCOM, *J. Geophys. Res.*, 114, C05011, doi:10.1029/2007JC004548

Chen C., R. Tian, R. C. Beardsley, J. Qi and Q. Xu, 2010. Modeling 2008 in Massachusetts Bay using an upgraded unstructured-grid Bays Eutrophication Model. Boston: Massachusetts Water Resources Authority. Report 2010-15. 127p.

Cheng, R. T., V. Casulli, and J. W. Gartner, 1993. Tidal, residual, intertidal mudflat (TRIM) model and its applications to San Francisco Bay, California. *Estuarine*, *Coastal* and *Shelf Science*, 36: 235-280.

Chorin, A. J., 1968. Numerical solution of Navier-Stokes equations. *Math Comput*, 22, 745-762.

Cockburn, B, S. Hou, and C. W. Shu, 1990. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, *Math Comp.*, **54**, 545-581.

Cotner, J. B. and R. G., Wetzel, 1992. Uptake of dissolved inorganic and organic phosphorus compounds by phytoplankton and bacterioplankton. *Limnol.&Oceanogr.*, **37**: 232-243.

Cowles, G, 2008. Parallelization of the FVCOM Coastal Ocean Model. *Int. J. High Performance Comput. Appl.,* 22 (2), 177-193.

Crank, J. and P. Nicolson, 1947. A practical method for numerical evaluation of partial differential equations of the heat conduction type. *Proceedings of the Cambridge Philosophical Society,* 43, 50-64.

Curry, J. A., J. L, Schramm and D. K. Perovich, 2001. Applications of SHEBA/FIRE data to evaluation of snow/ice albedo parameterizations, *J. Geophys. Res.*, 101:15,345-15,355.

Davis, C. S., 1987. Zooplankton life cycles. In Backus R. H. and D. W. Bourne, editors, *Georges Bank*, pages 256-267. MIT Press, Cambridge, Massachusetts.

Davis, A. M., J. E. Jones, and J. Xing. 1997. Review of recent developments in tidal hydrodynamic modeling, I: Spectral models. *J. Hydraulic Eng.*, ASCE, 123: 278-292.

Di Toro, D. M., D. J. O'Connor, and R. V. Thomann, 1971. A dynamic model of the phytoplankton population in the Sacramento-San Joaquin Delta. In: Nonequilibrium System in Natural Water Chemistry, Adv. Chem. Ser. 106, *Amer. Chem. Soc.*, Washington, DC, 131-180.

Di Toro, D. M. and J. J. Fitzpatrick, 1993. Chesapeake Bay sediment flux model. HydroQual, Inc., Mahwah, New Jesery.

Di Toro, D. M. and W. F. Matystik, 1980. Mathematical models of water quality in large Lakes. Part I: Lake Huron and Saginaw Bay, EPA-600/3-80-056, 28-30.

Di Toro, D. M. and J. P. Connolly, 1980. Mathematical models of water quality in large Lakes. Part II: Lake Erie, EPA-600/3-80-056, 90-101.

Donelan, M. A., 1990. Air-sea interaction. *The Sea*. B. LeNehaute and D. M. Hanes, eds., *Ocean Engineering Science*, **9**, Wiley and Sons, 239-292.

Donelan A.M., F.W. Dobson, S. D. Smith, and R. J. Anderson, 1993. On the dependence of sea surface roughness on wave development, *J. Phys. Oceanogr., 23*, 2143-2149, doi: 10.1175/1520-0485(2001)031.

Evensen, G., 1992. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. *J. Geophys. Res.*, 97(C11), 17905–19724.

Evensen, G., 1993, Open boundary condition for the extended Kalman filter with a quasi-geostrophic ocean model. *J. Geophys. Res.*, 98(C9), 16529–16546.

Evensen, G., 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.*, 99, 10,143-10,162.

Fahnenstiel, G. L. and D. Scavia, 1987. Dynamics of Lake Michigan phytoplankton: Recent changes in surface and deep communities. *Can. J. Fish Aquat. Sci.*, 44: 509-514.

Falgout, D. R. and U. M. Yang, 2002. H*ypre*: a library of high performance preconditioners. In: P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra and A.G. Hoekstra, Editors, *Computational Science – ICCS 2002 Part III*, *Lecture Notes in Computer Science* 2331, Springer-Verlag, pp. 632–641.

Fasham, M.J. R , H. W. Duklow, and S. M., 1990. A nitrogen-based model of plankton dynamics in the oceanic mixed layer. *J. Mar. Res.*, 48, 591-639.

Flather, R. A. and N. S. Heaps. 1975. Tidal computations for Morecambe Bay. *Geophys. J. Roy. Astr. Soc.*, 42:489-517.

Foreman, M. G. G., 1978. Manual for tidal analysis and prediction. Pacific Marine Science Rep. 78-6, Institute of Ocean Sciences, Patrcia Bay, Sydney, British Columbia, Canada, 70pp.

Forward, R. Jr., 1988. Diel vertical migration: zooplankton photobiology and behavior. *Oceanogr. Mar. Biol. Annu. Rev.*, 26, 361-392.

Frank, T. M. and E. A. Widder, 1997. The correlation of downwelling irradiane and staggered vertical migration patterns of zooplankton in Wilkinson Basin, Gulf of Maine, *J. Plankton Res.*, 19(12), 1975-1991.

Franks, P. J. S., J. S. Wroblewski and G. R. Flierl, 1986. Behavior of a simple plankton model with food-level acclimation by herbivores. *Mar. Biol*., *91*, 121-129.

Franks, P. J. S. and C. Chen, 1996. Plankton production in tidal fronts: a model of Georges Bank in summer. *J. Mar. Res*., **54**, 631-651.

Franks, P. J. S., and C. Chen, 2001. A 3-D prognostic numerical model study of the Georges Bank ecosystem. Part II: biological-physical model. *Deep Sea Res*. II, **48**, 457-482.

Fringer, O. B., M. Gerritsen and R. L. Street, 2006. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling*. 14, 139–173.

Galperin, B., L. H. Kantha, S. Hassid, and A. Rosati, 1988. A quasi-equilibrium turbulent energy model for geophysical flows. *J. Atmos. Sci*., 45, 55-62.

Gao, G., C. Chen, J. Qi, and R. C. Beardsley, 2011. An unstructured-grid, finite-volume sea ice model: Development, validation, and application, *J. Geophys. Res*., 116, C00D04, doi:10.1029/2010JC006688.

Ge, J**.**, C. Chen, J. Qi, P. Ding and R. C. Beardsley, 2012. A dike-groyne algorithm in a terrain-following coordinate ocean model (FVCOM): development, validation and application, Ocean Modelling, 47, 26-40. doi: 10.1016/j.ocemod. 2012.01.006.

Ghil, M. and P. Malanotte-Rizzoli, 1991. Data assimilation in meteorology and oceanography. *Advances in Geophysics,* Academic Press, 33, 141–266

Gismervik, I. and T. Andersen. 1997. Prey switching by *Acartia clausi*: experimental evidence and implications of intraguild predation assessed by a model. *Mar. Eco. Prog. Ser*., 157, 247-259.

Haidvogel, D. B, H. G. Arango, K. Hedstrom, A. Beckmann, P. M. Rizzoli and A. F. Schepetkin, 2000. Model evaluation experiments in the North Atlantic Basin, Simulation in nonlinear terrain-following coordinates. *Dyn. Atmo.Oceans*., 32, 239-281.

Hamilton, R. D. and J. E. Preslon, 1970. Observations on the continuous culture of a planktonic phagotrophic protozoan. *J. Exp. Mar. Biol*., **5**: 94-104.

Haurwitz, B., 1956. The geographical distribution of the solar semidiurnal pressure oscillation. *Meteorol. Pap*, 2.

Heggelund, Y., Vikebo, F., Bernsten, J and G. Furnes, 2004. Hydrostatic and non-hydrostatic studies of gravitational adjustment over a slope. *Cont. Shelf Res*. 24, 2133–2148.

Hervouet, J. M. and J. M. Janin. 1994. Finite-element algorithms for modeling flood propagation, p. 102-113. In P. Molinaro and L. Natale (eds.), *Modeling of Flood Propagation over Initially Dry Areas*, ASCE, New York

Hibler III, W. D., 1979. A dynamic thermodynamic sea ice model. *J. Phys. Oceanogr*. 9, 815–843.

Hofmann, E.E. and J.W. Ambler 1988. Plankton dynamics on the outer southeastern U.S. continental shelf. Part II: A time-dependent biological model. *J. Mar. Res*. 46:883–917.

Houtekamer, P. and H. L. Mitchell, 1998. Data assimilation using an ensemble Kalman filter technique. *Mon. Wea. Rev.*, 126, 796-811.

Hsu, T., S. Ou and J. Liau, 2005. Hindcasting nearshore wind waves using a FEM code for SWAN. *Coastal Engineering,* 52, 177-195.

Hubbard, M. E., 1999. Multidimensional slope limiters for MUSCL-type finite volume schemes on unstructured grids, *J. Comput. Phys.*, 155, 54-74.

Huang, H, C. Chen, G. W. Cowles, C. D. Winant, R. C. Beardsley, K. S. Hedstrom and D. B Haidvogel, 2008. FVCOM validation experiments: comparisons with ROMS for three idealized barotropic test problems. *J. Geophys. Res.*, 113, C07042, doi: 10.1029/2007JC004557.

Hunke, E. C. and W. H. Lipscomb, 2006. CICE: the Los Alamos Sea Ice Model Documentation and Software User's Manual, T-3 Fluid Dynamics Group, Los Alamos National Laboratory. Los Alamos, N. M.

Hunk, E. C. and J. K. Dukowicz., 1997. An elastic-viscous-plastic model for sea-ice dynam-dynamics. *J. Phys. Oceanogr*. 27, 1849–1867.

HydroQual, 2000. Bays Eutrophication Model (BEM): modeling analysis for the period 1992-1994. Boston: Massachusetts Water Resources Authority. Report 2000-02, 158pp.

HydroQual, 2004. "User's Guide for RCA, Release 3.0", Hydroqual, Inc, New Jersey.

HYPRE Team, 2001. HYPRE-high performance preconditions, user's manual. Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 48pp.

Ip, J. T. C., D. R. Lynch, and C. T. Friedrichs. 1998. Simulation of estuarine flooding and dewatering with application to Great Bay, New Hampshire. *Estuarine, Coastal and Shelf Science*, 47:119-141.

Ivlev, V. S., 1955. Experimental ecology of the feeding of fishes. Pischepromizdat, Moscow, Transl. from Russian by D. Scott, Yale University Press, New Haven, Connecticut, 1961

Janssen, P. A. E. M., 2001. Reply. *J. Phys. Oceanogr.*, 31, 2532-2544.

Jassby, A. D. and T. Platt, 1976. Mathematical formulation of the relationship between photosynthesis and light for phytoplankton. *Limnol&Oceanogr.*, 21(4), 540-547.

Ji, R., 2003. Biological and physical processes controlling the spring phytoplankton bloom dynamics on Georges Bank. Ph.D. Thesis. The University of Georgia, 216pp.

Jorgensen, S.E., Nielsen, S.N., Jorge, L.A., 1991. Handbook of Ecological Parameters and Ecotoxicology. Elsevier.

Josberger, E. G. and S. Martin, 1981. A laboratory and theoretical study of the boundary layer adjacent to a vertical melting ice wall in salt water, *J. Fluid Mech.,* 111,439-473.

Kanarska, Y. and V. Maderich , 2003. A non-hydrostatic numerical model for calculating free-surface stratified flows. *Ocean Dyn.* 53, 176–185.

Kanarska, Y., A. Shchepetkin and J. C. McWilliams, 2007. Algorithm for non-hydrostatic dynamics in the Regional Oceanic Modeling System. *Ocean Modelling.* 18, 143-174.

Kantha, L. and C. A. Clayson, 1994. An improved mixed layer model for geophysical applications. *J. Geophys. Res.*, 99 (C12), 25,235-25,266.

Kantha, L. and C. A. Clayson, 2000. Numerical models of oceans and oceanic processes. *International Geophysics*, 66, Academic Press, San Diego, CA 2000. 976pp.

Karypis, G. and Kumar, V. 1998. METIS* A Software Package for Partitioning Unstructured Graphs, partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. University of Minnesota, Department of Computer Science/ Army HPC Research Center Minneapolis.

Kawamiya, M.., M.. J. Kishi, Y. Yamanaka and N. Suginohara, 1995. An ecological-physical coupled model applied to station Papa. *J. Oceanogr.*, *51*, 635-664.

Kiefer, D.A. and B. G. Mitchell, 1983. A simple, steady state description of phytoplankton growth based on absorption cross section and quantum efficiency. *Limnol& Oceanogr.*, 28, 770-776.

Kim, T, R. G. Labiosa, T. Khangaonkar, Z. Yang, C. Chen, J. Qi and C. F. Cerco, 2010. "Development and evaluation of a coupled hydrodynamic (FVCOM) and water quality model (CE-QUAL-ICM)." In Proceedings of the 11th International Conference on Estuarine and Coastal Modeling, November 4-6, 2009, Seattle, Washington, ed. ML Spaulding, pp. 373-388. American Society of Civil Engineers, Reston, VA. doi:10.1061/41121(388)23

Kobayashi, M. H., J. M. C. Pereira and J. C. F. Pereira, 1999. A conservative finite-volume second-order-accurate projection method on hybrid unstructured grids. *J. Comput. Phys.*, 150, 40-45.

Kraus, E. B., 1972: Atmosphere-Ocean Interaction. *Clarendon Press*, 275pp.

Lai, Z., 2009. A non-hydrostatic unstructured-grid finite-volume coastal ocean model system (FVCOM-NH): development, validation and application. Ph.D. thesis. University of Massachusetts-Dartmouth, p.155.

Lai, Z., C. Chen, G. Cowles, and R. C. Beardsley, 2010. A Non-Hydrostatic Version of FVCOM, Part I: Validation Experiments, *J. Geophys. Res.-Oceans*, 115, doi:10.1029/2009 JC005525

Lai, Z., C. Chen, G. Cowles, and R. C. Beardsley, 2010. A Non-Hydrostatic Version of FVCOM, Part II: Mechanistic Study of Tidally Generated Nonlinear Internal Waves in Massachusetts Bay, *J. Geophys. Res.-Oceans*, doi: 10.1029/2010JC006331.

Lancelot, C., J. Staneva, D. van Eeckhout, J.M. Beckers and E. Stanev, 2002. Modelling the Danube influenced North-western continental shelf of the Black Sea II: Ecosystem response to changes in nutrient delivery by the Danube River after its damming in 1972. *Deep-Sea Res.* I, 54, 473-499.

Laws, E.A. and T.T. Bannister, 1980. Nutrient- and light-limited growth of Thalassiosira fluviatilis in continuous culture, with implications for phytoplankton growth in the ocean. *Limnol&Oceanogr.*, 25, 457-473.

Le Dimet, F., and O. Talagrand, 1986. Variational algorithm for analysis and assimilation of meteoro-logical observations: Theoretical aspects. *Tellus*, 38A, 97–110.

Leising, A. W. C. Gentleman and B. W. Frost, 2003. The threshold feeding response of microzooplankton within Pacific high-nitrate low-chlorophyll ecosystem model under steady and variable iron input. *Deep-Sea Res.* II, 50, 2877-2894. doi: 10.1016/j.dsr 2.2003.07.002.

Leonard, C. L., C. R. McClain, R. Murtugudde, E. E. Hafmann, and L. W. Harding Jr., 1999. An iron-based ecosystem model of the central equatorial Pacific. *J. Phys. Res.*, 104 (C1), 1325-1341.

Lipscomb, W. H., 2001. Remapping the thickness distribution in sea ice models, *J. Geophys. Res.*, 106, 13,989-14,000.

Lipscomb, W. H., E. C. Hunke, W. Maslowski, and J. Jakacki, 2007. Ridging, strength, and stability in high-resolution sea ice models, *J. Geophys. Res.*, 112, C03S91, doi:10.1029/2005JC003355.

Lorenc, A. C., 1981. A global three-dimensional multi-variated statistical interpolation scheme. *Monthly Weather Review*, 109, 701–721.

Lynch, D. R. and W. G. Gray. 1980. Finite element simulation of flow in deforming regions. *J. Computational Phys.*, 36: 135-153

Lynch, D. R., and C. E. Naimie, 1993. The M_2 tide and its residual on the outer banks of the Gulf of Maine, *J. Phys. Oceanogr.*, 23, 2222-2253.

Lyu, S. J., P. Malanotte-Rizzoli, D. McLaughlin and D. Enthekabi, 2005a. A comparison of data assimilation results from the deterministic and stochastic ensemble Kalman filters. Unpublished manuscript.

Lyu, S. J., P. Malanotte-Rizzoli, J. Hansen, D. Mclauglin and D. Entekhabi, 2005b. Optimal fixed and adaptive observation arrays in an idealized wind-driven ocean model. Unpublished manuscript.

Maday,Y. and A. T. Patera, 1988. Spectral element methods for the incompressible Navier-Stokes equations. In: State-of-the art surveys in computational mechanics, A. K. Noor, editor, ASME, New York.

Mahadevan, A., Oliger, J. and R. Street, 1996. A nonhydrostatic mesoscale ocean model. Part II: Numerical implementation. *J. Phys. Ocean*ogr, 26, 1881–1900.

Marshall, J., C. Hill, L. Perelman and A. Adcroft, 1997. A. hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *J. Geophys. Res.*; 102(C3): 5733–5752.

Matsuda, H, K. Kawasaki, N. Shigesada, E. Teramoto and L.M. Riccirdi,1986. Switching effect on the stability of the prey-predator system with three trophic levels. *J. Theo. Bio*. 122, 251-262.

Maykut, G. A., 1978. Energy exchange over young sea ice in the central Arctic, *J. Geophys. Res.*, 83, 3646-3658.

Maykut, G. and N. Untersteiner, 1971. Some result from a time-dependent, thermodynamic model of sea ice, *J. Geophys. Res.*, 76, 1550-1575.

Maykut, G. A. and D. K. Perovich, 1987. The role of shortwave in the summer decay of a sea ice cover. *J. Geophys. Res.*, 92, 7032–7044.

McPhee, M. G., 1992. Turbulent heat flux in the upper ocean under sea ice. *J. Geophys. Res.*, 9 7, 5365-5379.

Mellor, G. L. and A. Blumberg, 1985. Modeling vertical and horizontal diffusivities with the sigma coordinate system. *Mon. Wea. Res.*, 113, 1379-1383.

Mellor, G. L. and T. Yamada, 1982. Development of a turbulence closure model for geophysical fluid problem. *Rev. Geophys. Space. Phys.*, 20, 851-875.

Mellor, G. L. and A. Blumberg, 2004. Wave breaking and ocean surface layer thermal response. J. Phys. Oceanogr., 34, 693-698.

Moloney, C.L. and J. G. Filed, 1991. The size-based dynamics of plankton food webs. I. simulation model of carbon and nitrogen flows. *Journal of Planmton Research* 13, 1003-1038.

Morrow, R. A., and P. De Mey, 1995. Four-dimensional assimilation of altimetric and cruise data in the Azores current in 1992–1993. *J. Geophys. Res.*, 100(C12), 25,007–25,025.

Naimie, C. E., 1996. Georges Bank residual circulation during weak and strong stratification periods: prognostic numerical model results. *J. Geophys. Res*. 101(C3), 6469-6486.

Ohman, M. D., 1990. The demographic benefits of diel vertical migration by zooplankton. *Ecological Monographs*, 60, 257-281.

Olson, R. J., 1981. 15N tracer studies of the primary nitrite maximum. Journal of Marine Research 39, 203–226.

Orlanski, I., 1976. A simple boundary condition for unbounded hyperbolic flows. *J. Comput. Physics*, 21**,** 251-269.

Parsons, T. R., M. Takahashi, and B. Hargrave, 1984. *Biological Oceanographic Processes*, 3rd ed. Pergamon Press.

Pedlosky, J., 1974. Longshore currents, upwelling and bottom topography. *J. Phys. Oceanogr.*, 4, 214-226.

Peeters, J. C. H. and P. Eilers, 1978. The relationship between light intensity and photosynthesis, a simple mathematical model. Hydrobiological Bulletin 12, 134-136.

Perovich, D. K. and B. Elder, 2002. Estimates of ocean heat flux at SHEBA. *Geophys. Res. Let*. 29(9), 10.1029/2001GL014171.

Platt, T., C. L. Gallegos, and W. G. Harrison. 1980. Photoinhibition of photosynthesis in natural assemblages of marine phytoplankton. *J. Mar. Res*. 38, 687-701.

Pitchford, J.W. and J. Brindley, 1999. Iron limitation, grazing pressure and oceanic high nutrient-low chlorphyll (HNLC) regions. Journal of Plankton Research 21, 525-547.

Qi, J., C. Chen, R. C. Beardsley, W. Perrie, G. W. Cowles, and Z. Lai, 2009. An unstructured-grid finite-volume surface wave model (FVCOM-SWAVE): implementation, validations and applications. *Ocean Modelling*, 28, 153-166. doi:10.1016/j.ocemod.2009.01.007.

Reed, W. H., and T. R. Hill, 1973. Triangular and methods for the neutron transport equation, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory.

Rodi, W., 1980. Turbulence models and their application in hydraulics. Report., Int. Assoc. for Hydraul. Res., Delft, Netherlands.

Rogers, W. E., J. M. Kaihatu, H. A. H. Petit, N. Booij and L. H. Holthuijsen, 2002. Diffusion reduction in an arbitrary scale third generation wind wave model. *Ocean Engng.*, 29, 1357-1390.

Rosati, A. and K. Miyakoda, 1988. A general circulation model for upper ocean simulation, *J. Phys. Oceano.*, 18(11), 1601-1626, 1988.

Rothrock, D. A., 1975. The energetics of the plastic deformation of pack ice by ridging. *J. Geophys. Res.*, 80, 4514-4519.

Saad, Y., 2000. Iterative method for sparse linear systems. SIAM, P144-202.

Scavia, D., G. L. Fahnenstiel, M. S. Evans, D. L. Jude and J. T. Lehman, 1986. Influence of salmonine predation and weather on long-term water quality trends in Lake Michigan. *Can. J. Fish. Aquat. Sci*., 43: 435-443.

Scavia, D. and G. L. Fahnenstiel, 1987. Dynamics of Lake Michigan phytoplankton: mechanisms controlling epilimnetic communities. *J. Great Lake Res*., 13(2): 103-120.

Scavia, D., G. A. Lang and J. F. Kitchell, 1988. Dynamics of Lake Michigan plankton: a model evaluation of nutrient loading, competition, and predation. *Can. J. Fish. Aquat. Sci.*, 45: 165-177.

Sidén, G. L. D. and D. R. Lynch. 1988. Wave equation hydrodynamics on deforming elements. *International J. Num. Meth.* in *Fluids*, 8: 1071-1093.

Simpson, J. J. and T. D. Dickey, 1981a. The relationship between downward irradiance and upper ocean structure. *J. Phys. Oceanogr.*, 11, 309-323.

*Simpson, J. J. and T. D. Dickey, 1981b. Alternative parameterizations of downward irradianceand their dynamical significance. *J. Phys. Oceanogr.*, 11, 876-882.

*Smagorinsky, J., 1963. General circulation experiments with the primitive equations, I. The basic experiment. *Monthly Weather Review*, 91:99-164.

Smolarkiewicz, P. K., 1984. A fully multidimensional positive-definite advection transport algorithm with small implicit diffusion. *J. Comput. Phys.*, 54, 325-362.

Smolarkiewicz, P. K. and W. W. Grabowski, 1990. The multidimensional positive definite advection transport algorithm: nonoscillatory option. *J. Comput. Phys.*, 86, 355-375.

Smolarkiewicz, P. K., J. Szmelter, 2005. Multidimensional positive definite advection transport algorithm (MPD MPDATA): A): an edge-based unstructured-data formulation. *Int. J. Num. Meth. in Fluids* 47, 1293–1299.

Smith, S. D., R. J. Anderson, W. A. Oost, C. Kraan, N. Maat, J. DeCosmo, K. B. Katsaros, K. L. Davidson, K. Bumke, L. Hasse, and H. M. Chadwich, 1992. Sea Surface Wind Stress and Drag Coefficients: The HEXOS Results. *Boundary-Layer Meteorol.*, 60, 109-142.

Spitz, Y.H., Moisan, J.R. and M. R. Abbott, 2001. Configuring an ecosystem model using data from the Bermuda Atlantic Time Series (BATS). Deep-Sea Research Part II 48, 1733-1768.

Stacey, M. W., 1999. Simulation of the wind-forced near-surfacee circulation in Knight Inlet: a parameterization of the roughness length. *J. Phys. Oceanogr.*, 29, 1363-1367.

Stansby, P. and J. Zhou, 1998. Shallow-water flow solver with non-hydrostatic pressure: 2D vertical plane problem. *Int. J. Num. Meth. in Fluids* 28, 541–563.

Steele, J. H., 1962. Environmental control of photosynthesis in the sea. *Limnol&Oceanogr.*, 7, 137-150.

Steele*, M. 1992. Sea ice melting and floe geometry in a simple ice-ocean model. Journal of Geophysical Research,* 97(C11):17,729-17,738.

Steele, J. H. and E. W. Henderson, 1992. The role of predation in plankton models. *J. Plankton Res.*, *14*, 157-172.

Strom, S. L. and H. Loukos, 1998. Selective feeding by protozoan: model and experimental behaviors and their consequences for populations stability. Journal of Plankton Research 20, 831-846.

SWAN Team, 2006a. SWAN Cycle III version 40.51 Technical documentation. Delft University of Technology, Faculty of Civil Engineering and Geosciences, Environmental Fluid Mechanics Section, P.O. Box 5048, 2600 GA Delft, The Netherlands.

SWAN Team, 2006b. SWAN Cycle III version 40.51 user manual. Delft University of Technology, Faculty of Civil Engineering and Geosciences, Environmental Fluid Mechanics Section, P.O. Box 5048, 2600 GA Delft, The Netherlands.

SWAMP Group, 1985. *Ocean Wave Modelling*. Plenum, New York, 256 pp.

Tamiya, H., E. Hse, K. Shibata, A. Mituya, T. Iwamura, T. Nihei and T. Sasa, 1985. Kinetics of growth of chlorella, with special reference to its dependence on quality of available light and on temperature. In. Algal Culture: from Laboratory to Pilot Plat. J. S. Berlo (Ed.) Carneigie Inst. Publication 600. pp. 204-232.

Tansky, 1978.

Terray, E. A., M. A. Donelan, Y. C. Agrawal, W. M. Drnnan, K. K. Kahma, A. J. Williams, P. A. Hwang, and S. A. Kitaigorodski, 1996. Estimates of kinetic energy dissipation under breaking waves. *J. Phys. Oceanogr.*, 26, 792-807.

Terray, E. A., M. A. Donelan, Y. C. Agrawal, W. M. Drnnan, K. K. Kahma, A. J. Williams, P. A. Hwang, and S. A. Kitaigorodski, 1997. Reply. *J. Phys. Oceanogr.*, 27, 2308-2309.

Terray, E. A., W. M. Donelan, and M. A. Donelan, 2000. The vertical structure of shear and dissipation in the ocean surface layer. Proc. Symp. On Air-Sea Interaction Sdyney, Australia, University of New South Wales, 239-245.

Thacker, W. C. and R. B. Long, 1988. Fitting dynamics to data. *J. Geophys. Res.*, 93(C2), 1227–1240.

Thomann, R. V., D. M. Di. Toro, and D. J. O'Connor, 1974. Preliminary model of Potomac estuary phytoplankton, *J. Env. Engr. Div.* ASCE 100: 699-708.

Thomann, R. V. and J. J. Fitzpatrick, 1982. Calibration and verification of a mathematical model of the eutrophication of the Potomac estuary. HydroQual, Inc., Mahwah, New Jersey.

Thorndike, A. S., D. S. Rothrock, G. A. Maykut and R. Colony, 1975. The thickness distribution of sea ice, *J. Geophys. Res.*, 80, 4501-4513.

Tian, R.C., A. F. Vézina, M. Starr and F. Saucier, 2001. Seasonal dynamics of coastal ecosystems and export production at high latitudes: a modeling study. *Limnol&Oceanogr.* 46, 1845-1859.

Tilman, D., S. Kilham and P. Kilham, 1982. Phytoplankton community ecology: the role of limiting nutrients. *Annu. Rev. Ecol. Syst.*, 13: 349-372.

Tippett, M. K., J. L. Anderson, C. H. Bishop, T. M. Hamill and J. S. Whitaker, 2003. Ensemble square root filters. *Mon. Wea. Rev.*, 131, 1485-1490.

Tolman, H. L., 2002. User manual and system documentation of WAVEWATCH-III version 2.22. Environmental Modeling Center, Marine Modeling and Analysis Branch, NOAA.

Tremblay, B. and L. A. Mysak, 1997. Modeling sea ice as a granular material, including the dilatancy effect. *J. Phys. Oceanogr.* 27, 2342–2360.

Totterdell, I. J., 1993. An annotated bibliography of marine biological models. In Evan, G. T. and Fasham, M. J. R. (eds), towards a model of ocean biogeochemical processes. Springer-Verlag-Berlin, pp 317-339.

Tziperman, E. and W. C. Thacker, 1989. An optimal-control/adjoint-equations approach to studying the oceanic general circulation. *J. Phys. Oceanogr.*, 19, 1471–1485.

Utke, J., U. Naumann, M. Fagan, N. Tallent, M. Strout, P. Heimbach, C. Hill and C. Wunsch, 2008. OpenAD/F: A modular open-source tool for automatic differentiation of Fortran Codes. ACM: transactions on mathematical software 34(4), Article 18, http://doi.acm.org/10.1145/1377596.1377598.

Valiela, I., 1995. Marine Ecological Processes. Springer-Verlag New York. 686pp.

Vance, R. R., 1978. Predation and resource partitioning in one-predator, two-prey model communities. American Naturalist 112, 797-813.

Verity, P. G., 1991. Measurement and simulation of prey uptake by marine planktonic ciliates fed plastic and aplastic nanoplankton. *Limnol&Oceanogr.*, 36, 729-750.

Vollenweider, R. A., 1965. Calculation models of photosynthesis-depth curves and some implications regarding day rate estimates in primary production measurements. Memorie dell'instituto italiano di idrobiologia (Mem. Ist. Ital. Idrobiol) 18, 425-457.

WAMDI Group, 1988. The WAM model – A third generation ocean wave prediction model, *J. Phys. Oceanogr.*, 18, 1775-1810.

Wang, X., and C. H. Bishop, 2003. A comparison of breeding and ensemble transform Kalman filter ensemble forecast schemes. *J. Atmos. Sci.*, 60, 1140-1158.

Warner, J. C., C. R. Sherwood, R. P. Signell, C. Harris and H. G. Arango, 2008. Development of a three-dimensional, regional, coupled wave, current, and sediment-transport model, *Comp. & Geos., 34*(10), 1284-1306, doi: 10.1016/j.cageo.2008.02.012.

Webb, W.L., M. Newton and D. Starr, 1974. Carbon dioxide exchange of alnus rubra: a mathematical model. *Oecologi*a 17, 281-291.

Whitaker, J. S. and T. M. Hamill, 2002. Data assimilation without perturbed observations. *Mon. Wea. Rev.*, 130, 1913-1924.

Wilcox, D, 2000. Turbulence modeling for CFD, DCW Industries, Inc. 540pp.

Wu, L. and D. B. Bogy, 2000. Use of an upwind finite volume method to solve the air bearing problem of hard disk drives. *Comput. Mech.*, 26, 592-600.

Wu, L, C. Chen, F. Guo, M. Shi, J. Qi, and J. Ge, 2011. A FVCOM-based unstructured grid wave, current, sediment transport model, I. model description and validation, *J. Ocean. Univ. China*, 10(1), 1-8, doi: 10.1007/s11802-011-1788-3.

Xue, P., C. Chen, R. C. Beardsley, and R. Limeburner, 2011. Observing system simulation experiments with ensemble Kalman filters in Nantucket Sound, Massachusetts, *J. Geophys. Res.*, 116, C01011, doi:10.1029/2010JC006428.

Xue, P., C. Chen, and R. C. Beardsley, 2012. Observing system simulation experiments of dissolved oxygen monitoring in Massachusetts Bay, *J. Geophys. Res.*, 117, C05014, doi:10.1029/2011JC007843

Yossef, M. F. M. and H. J. Vriend, 2011. Flow details near river groynes: experimental investigation. *J. Hydraulic Eng.*, 13, 504-516.

Zang, X. and P. Malanotte-Rizzoli, 2003. A comparison of assimilation results from the Ensemble Kalman filter and the Reduced-Rank Extended Kalman filter. *Nonlinear Processes in Geophysics*, 10, 6,477-6,491.

Zhang, J. and W. D. Hibler III, 1997. On an efficient numerical method for modeling sea ice dynamics. *J. Geophys. Res*. 102, 8691–8702.

Zhao, L., R. Tian, P. Xue, C. Chen, W. S. Leo and M. J. Mickelson, 2011. Modeling 2010 in Massachusetts Bay using the unstructured-grid Bays Eutrophication Model. Boston: Massachusetts Water Resources Authority. Report 2011-09. 118 p.

Zheng, L, C. Chen, and H. Liu, 2003a. A modeling study of the Satilla River Estuary, Georgia. Part I: flooding/drying process and water exchange over the salt marsh-estuary-shelf complex. *Estuaries*, 26 (3), 651-669.

Zheng, L. C. Chen, M. Alber, and H. Liu, 2003b. A modeling study of the Satilla River Estuary, Georgia. II: Suspended sediment. *Estuaries*, 26(3), 670-679.

Zheng, L. C. Chen, and F. Zhang, 2004. Development of water quality model in the Satilla River estuary, Georgia. *Ecological Modeling*, 178, 457-482.

Zijlema, M. and G. S. Stelling, 2005. Further experiences with computing non-hydrostatic free-surface flows involving water waves. *Int. J. Numer. Methods Fluids* 48(2), 169-197.